DATA MINING VIA SUPPORT VECTOR MACHINES: SCALABILITY, APPLICABILITY, AND INTERPRETABILITY

BY

HWAN-JO YU

B.A., Chung-Ang University, Seoul, 1997

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

ABSTRACT

Data is everywhere, abundant, continuously increasing, and heterogeneous. For example, Web pages on the Internet are rapidly growing, data in commercial databases are continuously accumulating, as well as biological data (e.g., microarray data), scientific data (e.g., astronomical data), and text data (e.g., news articles and science journals). Extracting useful information or knowledge from such massive data is important but hard: rich data but poor information is a common phenomenon in the real world. Data mining refers to extracting or mining useful knowledge from large amounts of data. Data mining includes the tasks of data classification, clustering or outlier analysis, concept description or visualization, association analysis, and evolution or trend analysis [26].

Many statistical and machine learning methods have been employed for data mining subtasks. Based on a strong mathematical foundation, SVMs (Support Vector Machines) have been one of the most actively developed classification and regression methodologies due to their salient properties such as margin maximization and systematic nonlinear classification via kernel tricks. However, SVMs have not been as favored for large-scale data mining because of several challenges:

1. *Scalability:* SVMs are unscalable to data size while common data mining applications often involve millions or billions of data objects.

2. *Applicability:* SVMs are limited to (semi-)supervised learning which is mostly applied to binary classification problems.

3. *Interpretability:* It is hard to interpret and extract knowledge from SVM models.

In this thesis, these challenges are identified and the SVM technologies are advanced to the practice of data mining with the proposal of several practical data mining solutions as follows.

1. A scalable SVM classification method, called CB-SVM (Clustering-Based SVM), is proposed. CB-SVM effectively approximates an SVM classification function in a single scan of data given a fixed size of memory [57]. CB-SVM applies a hierarchical micro-clustering method that provides an SVM with high quality samples that carry the statistical summaries of the data such that the summaries maximize the benefit of learning the SVM.

2. Two single-class classification (SCC) methods are developed using SVMs. SCC seeks to distinguish one class of data from the universal set of multiple classes. (We call the target class *positive* and the complement set of samples *negative*.) In SCC problems, it is assumed that a reasonable sample of the negative data is not available. Since it is not natural to collect the "non-interesting" objects (i.e., negative data) to train the concept of the "interesting" objects (i.e., positive data), SCC problems are prevalent in real world where positive and unlabeled data are widely available but negative data are hard or expensive to acquire. Two SCC algorithms – *Mapping Convergence (MC)* and *Support Vector Mapping Convergence (SVMC)* – are proposed that compute an accurate boundary of the target class from positive and unlabeled data (without labeled negative data) [55].

3. Finally, SVM is employed to identify compact and highly discriminative feature combinations for a specified class of data by directly analyzing the SVM model of polynomial kernel. Nowadays in bioinformatics, high throughput techniques such as microarray experiments make it feasible to examine and collect massive data at the molecular level. These data, typically mapped to a very high dimensional feature space, carry rich information about functionalities of certain biological entities and can be used to infer valuable knowledge for the purposes of classification and prediction. Typically, a small number of features or feature combinations play determinant roles in functional discrimination. The identification of such features or feature combinations is of great importance. A method is presented that employs SVM to discover compact and highly discriminative features or feature combinations from a rich feature collection [58].

Our results show that, by appreciating the SVM's prominent properties, the proposed methods are shown better than other existing methods in many real-world data mining problems.

To my parents

ACKNOWLEDGMENTS

I am deeply thankful to my advisors Prof. Jiawei Han for his insightful guidance, enlighting advice, and endless encouragement throughout my PhD study. Without his help and support, this thesis would likely never have seen the light of day. I feel exceedingly privileged and honored to be a student of his. I would also like to thank a number of faculties who have influenced my time and study in this school - Prof. Lenny Pitt, Prof. Sylvian Ray, Prof. ChengXiang Zhai, Prof. Kevin Chang, Prof. AnHai Doan, and Prof. Jiong Yang.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

Introduction

Data is everywhere, abundant, continuously increasing, and heterogeneous. For example, Web pages on the Internet are rapidly growing, data in commercial databases are continuously accumulating, as well as biological data (e.g., microarray data), scientific data (e.g., astronomical data), and text data (e.g., news articles and science journals). Extracting useful information or knowledge from such massive data is important but hard: rich data but poor information is a common phenomenon in the real world. Data mining refers to extracting or mining useful knowledge from large amounts of data. Data mining may include but not limited to the following sub-tasks. ([26] provides an excellent summary of data mining tasks.)

1. **Concept/Class Description and Visualization** includes (1) data characterization, by summarizing or visualizing the data of the class under study (often called the target class) in general terms, or (2) data discrimination, by comparison of the target class with one or a set of comparative classes (often called the contrasting classes).

2. **Association Analysis** is the discovery of association rules showing relations between attribute-value conditions. For instance, given the Walmart database, a data mining system may find association rules such as "people who buy beers also buy diapers".

3. **Classification and Prediction** is the process of finding a model that describes and distinguishes data classes or concepts, for the purpose of predicting the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known). Classification learns from training data and builds a model that discriminates fraud examples from normal cases.

Those models or functions may be simple like collections of specific patterns or complicated involving multiple correlations of invisible factors.

4. **Cluster analysis** is to group the similar data objects based on some similarity measure such that the intraclass similarity is maximized and interclass similarity is minimized. Each cluster, a group of similar data, can be viewed as a class of objects. Clustering can also facilitate taxonomy formation, that is, the organization of observation into a hierarchy of classes that group similar events together.

5. **Outlier analysis** detects data that do not comply with the general behavior or model of the data. Most data mining methods discard outliers as noise or exceptions. However, in some other applications, the rare events can be more interesting than the more regularly occurring ones. For example, outlier analysis may uncover fraud by detecting purchases of extremely large amounts for a given account number in comparison to regular charges incurred by the same account.

6. **Evolution Analysis** describes and models regularities or trends for data whose behavior changes over time.

Many statistical and machine learning methods have been employed for the data mining subtasks. Recently, SVMs (Support Vector Machines) have been one of the most actively developed classification and regression methodologies in machine learning community [51, 10, 30, 12, 46] due to their salient properties:

- **Margin maximization**: The classification boundary function of SVMs maximize the margin, which, in machine learning theory, corresponds to maximizing the *generalization* performance. (See Section 2.2 for more details.)

- **Nonlinear transformation of the feature space using the kernel trick**: SVMs efficiently handle nonlinear classifications using the kernel trick which implicitly transforms the input space into another high dimensional feature space.

The success of SVMs in machine learning naturally leads to its possible extension to large-scale data mining. However, SVMs have not been as favored for large-scale data mining because of several challenges:

1. **Scalability:** SVMs are unscalable to data size while common data mining applications often involve millions or billions of data objects.

2. **Applicability:** SVMs are limited to (semi-)supervised learning which is mostly applied to binary classification problems. However, data mining includes other tasks beyond binary classification

3. **Interpretability:** It is hard to interpret and extract knowledge from SVM models.

In this thesis, we address these challenges by introducing several practical data mining problems: (1) classifying from large data sets via SVMs (which addresses the challenge of scalability), (2) single-class classification (classification without negative data) via SVMs (which addresses the challenge of applicability), and (3) discovering compact and highly discriminative features or feature combination via SVMs (which addresses the challenges of applicability and interpretability). These three problems will be dealt with at Chapters 2, 3, and 4 respectively. The following section shows the detailed organization of the thesis. At the end of this chapter, we will explicate the primary novel contributions of the thesis.

1.1 Organization

In Chapter 2, we first tackle the problem of scaling SVMs to large data sets. The training complexity of SVMs is highly dependent on the size of a data set. Many real-world data mining applications involve millions or billions of data records where multiple scans of the entire data are too expensive to perform. Chapter 2 presents a method called *Clustering-Based SVM (CB-SVM)* which is specifically designed for handling very large data sets [57]. CB-SVM applies a hierarchical micro-clustering algorithm that scans the entire data set only once to provide an SVM with high quality samples that carry the statistical summaries of the data such that the summaries maximize the benefit of learning the SVM. CB-SVM uses the hierarchical cluster as an index of SVMs in order to maximize the SVM performance given limited amount of system resources (i.e., memory). Our analyses show that the training complexity of CB-SVM is quadratically depending on the number of support vectors, which is usually much smaller than that of entire data. Our experiments on synthetic and real data sets show that CB-SVM

is highly scalable for very large data sets and at the same time leads to high classification accuracy.

In Chapter 3, we deal with Single-Class Classification (SCC) via SVMs. Single-Class Classification (SCC) seeks to distinguish one class of data from the universal set of multiple classes. (We call the target class *positive* and the complement set of samples *negative*.) In SCC problems, it is assumed that a reasonable sample of the negative data is not available. Since it is not natural to collect the "non-interesting" objects (i.e., negative data) to train the concept of the "interesting" objects (i.e., positive data), SCC problems are prevalent in real world where positive and unlabeled data are widely available but negative data are hard or expensive to acquire. We present an SCC algorithm called *Mapping Convergence (MC)* that computes an accurate boundary of the target class from positive and unlabeled data (without labeled negative data) [55]. The basic idea of MC is to exploit the natural "gap" between positive and negative data by incrementally labeling negative data from the unlabeled data using the *margin maximization* property of SVM. We also present a method called *Support Vector Mapping Convergence (SVMC)* which optimizes the MC algorithm for fast training. Our analyses show that MC and SVMC without labeled negative data significantly outperform other SCC methods and generate as accurate boundaries as standard SVM with fully labeled data when the positive data is not very under-sampled and there exist gaps between positive and negative classes in the feature space. Our results also show that SVMC trains much faster than MC with very close accuracy.

Another challenging problem in using SVMs for data mining is to understand the SVM model for further analysis and mining knowledge from the data. Chapter 4 presents a method that analyzes the SVM model to extract useful information, with an application of it in bioinformatics [58]. Nowadays, high throughput techniques such as microarray experiments make it feasible to examine and collect massive data at the molecular level. These data, typically mapped to a very high dimensional feature space, carry rich information about functionalities of certain biological entities and can be used to infer valuable knowledge for the purposes of classification and prediction. Typically, a small number of features or feature combinations play determinant roles in functional discrimination. The identification of such features or feature combinations is of great importance. In this chapter, we study the problem of discovering compact and highly discriminative features or feature combinations from a rich feature collection. We employ the support vector machine as the classification means and aim at finding com-

pact feature combinations. Comparing to previous methods on feature selection, which identify features solely based on their individual roles in the classification, our method can identify minimal feature combinations that ultimately have determinant roles in a systematic fashion. Experimental study on drug activity data shows that our method can discover descriptors that are not necessarily significant individually but are most significant combinatively.

Chapter 5 concludes the thesis by discussing the various results presented and giving directions for future research. This thesis is a step in the direction towards developing solid and systematic methodology for practical data mining problems. A lot of interesting problems still remain unsolved and will be the focus of future research.

1.2 Contributions

Original contributions presented in this thesis span the areas of machine learning and data mining algorithms, and the applications of intrusion detection, text processing, pattern recognition, and bioinformatics. In particular, some of the applications and algorithms addressed in this thesis are the followings.

- We address in Chapter 2 the problem of classifying abnormal network data from a large network data stream. We used the network intrusion detection data set from the KDD Cup 1999, which consists of about five millions of training and three hundred thousands testing data records. Any existing SVM implementations would generate system failure due to memory blowup or would take "forever" to finish training since their training time is high depending on the data size. The CB-SVM algorithm we propose efficiently trains a classification function from it and classifies the abnormal data packets with high accuracy. CB-SVM is more effective on very large data sets having different probability distributions of training and testing data on which random sampling usually hurts. The network intrusion data set is a good example of such cases because the testing data is not from the same probability distribution as the training data, and it also includes specific attack types not in the training data. This is because they were collected in different times of periods, which makes the task more realistic. Currently, CB-SVM does not perform well on high-dimensional data due to the limitation of the clustering method it is based

on. Developing a high-dimensional indexing structure for SVMs is an important direction of future work as we will discuss in Chapter 5.

- In text document or Web page classifications, classifying the documents or Web pages of interest is a common task, for instance, to automatically classify the interesting documents from all the others or to collect the interesting Web pages from the Internet. In these scenarios of Single-Class Classification (SCC), it is often hard to acquire a reasonable sample of the negative data. For example, in text or Web page classification (e.g., personal homepage classification), collecting negative training data (e.g., a sample of "non-homepages") is delicate and arduous because manually collected negative data could be easily biased because of a person's unintentional prejudice, which could be detrimental to classification accuracy. The MC and SVMC algorithms we propose in Chapter 3 learn accurate classification functions from only the samples of positive and unlabeled data, without negatively labeled data. Our proposed algorithms are general and applicable to other applications in which labeling data is hard or expensive, such as (1) automatic diagnosis of diseases where it is expensive to perform a detection test to collect negative data, (2) pattern recognition where collecting an unbiased sample of negative patterns is hard. We perform the experiments on the data sets of text & Web documents, letter recognition, and breast cancer classification. MC and SVMC significantly outperformed all the other SCC methods we compared with in most cases.

- We proposed in Chapter 4 an algorithm that extracts discriminative feature combinations from a rich feature space. Experimental study on drug activity data shows that our method can discover descriptors that are not necessarily significant indivisually but are most significant combinatively. It also identifies a much smaller set of features that produce a better classification quality than the (larger) set of features selected by other methods. We believe that our proposed method is also applicable to other biological data such as gene expression data and protein subcellular location data.

This thesis focuses on the above data mining applications. However, we believe that the algorithms presented in the thesis can be further applied to other applications of related areas including speech recognition, computer vision, and natural language processing.

CHAPTER 2

Making SVM Scalable to Large Data Sets using Hierarchical Cluster Indexing

2.1 Introduction

Support vector machines (SVMs) have been promising methods for data classification and regression [51, 10, 30, 12, 46, 56]. Their successes in practice are drawn by their solid mathematical foundations which convey several salient properties:

- **Margin maximization**: The classification boundary functions of SVMs maximize the margin, which, in machine learning theory, corresponds to maximizing the *generalization* performance. (See Section 2.2 for more details.)
- **Systematic nonlinear classification via kernel tricks**: SVMs efficiently handle nonlinear classifications using kernel tricks, which implicitly transforms the input space into a high dimensional feature space.

The success of SVMs in machine learning naturally leads to its possible extension to large-scale data mining. However, despite the prominent properties of SVMs, they are not as favorably used for large-scale data mining as for pattern recognition or machine learning because the training complexity of SVMs is highly dependent on the size of a data set. (It is known to be at least quadratic to the number of data points. Refer to [12] for more discussions on the complexity of SVMs.) Many real-world data mining applications often involve millions or billions of data records. For such large data sets, existing SVM implementations (1) generate system failures due to memory blowup, or (2) take "forever" to finish training with unbounded memory. (In our experiments in Section 2.5, two efficient SVM implementations, LIBSVM and Active SVM, generated system failures for large data sets.)

Researchers have proposed various revisions of SVMs to increase the training efficiency by mutating or approximating it. However, they are still not feasible with very large data sets where even multiple scans of the entire data set are too expensive to perform. (See Section 2.7 for the discussions on related work.)

This paper presents a new approach for scalable and reliable SVM classification, called *Clustering-Based SVM (CB-SVM)*, which is specifically designed for handling very large data sets given a limited amount of system resource, e.g., memory. When the size of the data set is large, SVMs tend to perform worse with training from the entire data than training from a fine quality of samples [42]. The SVM-based selective sampling (or active learning) techniques try to select the training data intelligently to maximize the performance of SVM, but they normally require many scans of data set [42, 49] (Section 2.7). Our CB-SVM applies a hierarchical micro-clustering algorithm that scans the entire data set only once to provide an SVM with high quality "*samples*" that carry the statistical summaries of the data such that the summaries maximize the benefit of learning the SVM. CB-SVM is scalable in terms of the training efficiency while maximizing the performance of SVM with bounded memory. The hierarchical micro-clusters conceptually play the role of indexing an SVM. By using the hierarchical micro-clustering algorithm called *BIRCH* [59], CB-SVM also automatically controls the trade-off between the memory size and the accuracy of SVM.

The key idea of CB-SVM is to use the hierarchical micro-clustering technique to get finer description closer to the boundary and coarser description farther from the boundary, which can be efficiently processed as follows: CB-SVM first constructs two micro-cluster trees called *CF-tree* from positive and negative training data respectively. In each tree, a node in a higher level is a summarized representation of its children nodes. After constructing the two trees, CB-SVM starts training an SVM only from the root nodes. Once it generates the "rough" boundary from the root nodes, it selectively declusters only the data summaries near the boundary into lower (or finer) levels using the tree structure. The hierarchical representation of the data summaries is a perfect base structure for CB-SVM to effectively perform the selective declustering. CB-SVM will repeat this selective declustering down to the leaf level.

CB-SVM can be used for classifying very large data sets of relatively low dimensions, such as streaming data or data in large data warehouses. It especially performs well where random sampling hurts the performance due to infrequently occurring important data or irregular pat-

terns of incoming data, which results in different probability distributions between training and testing data. We discuss this more in Section 2.5.1.3. Our experiments on the network intrusion data set (Section 2.5.2), a good example which shows that random sampling could hurt, show that CB-SVM is scalable for very large data sets while also generating high classification accuracy. Due to the limitation of clustering-based indexing structure, CB-SVM performs not as well for high dimensional data. Developing an indexing structure for high dimensional data is an important future work.

To the best of our knowledge, the proposed method is currently the only SVM for very large data sets which tries to generate the best results with bounded memory.

The remainder of the paper is organized as follows. We first overview SVM in Section 2.2. In Section 2.3, we introduce a hierarchical micro-clustering algorithm for very large data sets, originally exploited by T. Zhang et al. [59]. In Section 2.4, we present the CB-SVM algorithm that applies the hierarchical micro-clustering algorithm to a standard SVM to make the SVM scalable for very large data sets. Section 2.5 demonstrates experimental results on artificial and real data sets. Section 2.6 presents the extension of CB-SVM to the SVM nonlinear kernels. We discuss the related work in Section 2.7 and conclude our study in Section 2.8.

2.2 SVM Overview

In machine learning theory, given a limited number of training data set $\{(x, y)\}$ (y is the label of x), the optimal class boundary function (or hypothesis) $h(x)$ is considered as the one that gives the best *generalization* performance on "*unseen*" examples rather than on the training data. The performance on the training data is not regarded as a good evaluation measure for a hypothesis because the hypothesis ends up *overfitting* when it tries to fit the training data too well. When a problem is easy to classify and the boundary function is more complicated than it needs to be, the boundary is likely *overfit*. When a problem is hard and the classifier is not powerful enough, the boundary is likely *underfit*. SVM is an excellent example of supervised learning that tries to maximize the generalization by maximizing the *margin* and also supports nonlinear separation using kernel tricks, by which SVM tries to avoid overfitting and underfitting [10, 51]. The *margin* in SVM denotes the distance from the boundary to the closest data points in the feature space as illustrated in Figure 2.1: In the figure, both the left

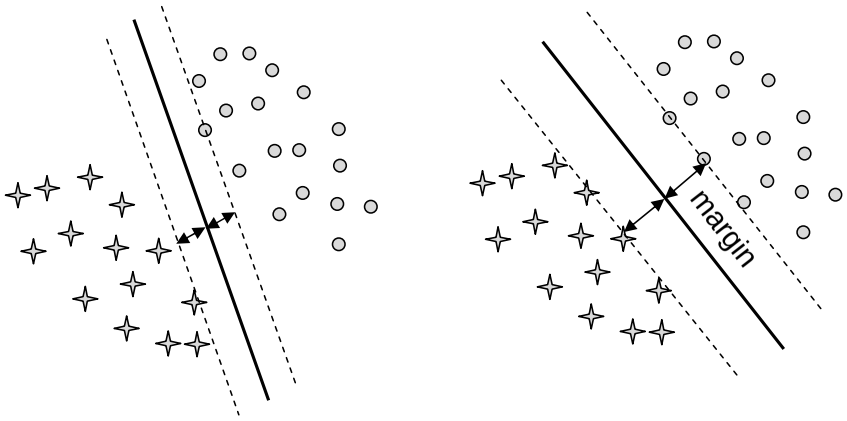


Figure 2.1 Margin in SVM

and right classification functions correctly classify the training data. However, intuitively, the right figure with a larger margin makes the two groups more distinguishing for unseen data, i.e., has a higher generalization performance. SVM computes this boundary of the largest margin in feature space.

In SVM, the problem of computing a margin maximized boundary function is specified by the following quadratic programming (QP) problem:

$$\begin{aligned} &minimize: \quad W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \tfrac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \\ &subject\ to: \qquad \sum_{i=1}^{l} y_i \alpha_i = 0 \\ &\qquad\qquad\qquad\qquad \forall i : 0 \leq \alpha_i \leq C \end{aligned}$$

The number of training data is denoted by l, α is a vector of l variables, where each component α_i corresponds to a training data (x_i, y_i), and C is the soft margin parameter controlling the influence of the outliers (or noise) in training data.

The kernel $k(x_i, x_j)$ for linear boundary function is $x_i \cdot x_j$, a scalar product of two data points. The nonlinear transformation of the feature space is performed by replacing $k(x_i, x_j)$ with an advanced kernel, such as polynomial kernel $(x^T x_i + 1)^p$ or RBF kernel $exp(-\frac{1}{2\sigma^2}||x - x_i||^2)$. The use of an advanced kernel is an attractive computational short-cut, which forgoes an expensive creation of a complicated feature space. An advanced kernel is a function that operates on the input data but has the effect of computing the scalar product of their images in a usually much

higher-dimensional feature space (or even an infinite-dimensional space), which allows one to work implicitly with hyperplanes in such highly complex spaces.

Another characteristic of SVM is that its boundary function is described by the *support vectors* (SVs) which are the data located the closest to the boundary. The above QP problem computes a vector α, each element of which specifies the weight of each data, and the SVs is the data whose corresponding α is greater than zero. In other words, the other data rather than the SVs do not contribute to the boundary function at all, and thus computing an SVM boundary function can be viewed as finding the SVs with the corresponding weights.

There have been many attempts to revise the original QP formulation such that it can be solved by a QP solver more efficiently [39, 19, 1]. (See Section 2.7 for more details.) We do not revise the original QP formulation of SVM. Instead, we try to provide a smaller but high quality data set that is beneficial to effectively computing the SVM boundary function by applying a hierarchical clustering algorithm. Our CB-SVM algorithm substantially reduces the total number of data points for training SVM while trying to keep the high quality of SVs that best describes the boundary.

2.3 Hierarchical Micro-Clustering Algorithm for Large Data Sets

The hierarchical micro-clustering algorithm we present here and will apply to our CB-SVM in Section 2.4 was originally exploited by T. Zhang et al. [59] and is called *BIRCH*. It introduces the concept of "micro-cluster" [59], denoting a statistically summarized representation of a group of data which are so close together that they are likely to belong to the same cluster. Our hierarchical micro-clustering algorithm has the following characteristics.

- It constructs a micro-cluster tree, called *CF (Clustering Feature) tree*, in one scan of the data set given a limited amount of resources (i.e., available memory and time constraints) by incrementally and dynamically clustering incoming multi-dimensional data points. Since the single scan of data does not allow backtracking, localized inaccuracies may exist depending on the order of data input. However, the CF tree captures the major distribution patterns of the data and provides enough information for CB-SVM to perform.

- It handles noise or outliers (data points that are not part of the underlying distribution) effectively as a by-product of the clustering.

Other hierarchical clustering algorithms also have been developed including STING [52], CURE [24], and Chameleon [33]. STING is a grid-based clustering method in which the spatial data is divided with equal space without considering much the data distribution. Chameleon has shown to be powerful at discovering arbitrarily shaped clusters of high quality, but its complexity in the worst case is $O(n^2)$, where n is the number of data points. CURE produces high-quality clusters with complex shapes, and its complexity is also linear to the number of objects, however, its parameter setting in general has a significant influence on the results. The CF tree of BIRCH carries the spherical shapes of hierarchical clusters and captures the statistical summaries of the entire data set. Thus it provides an efficient and effective structure for CB-SVM to run.

2.3.1 Clustering Feature and CF Tree

First, we define some basic concepts. Given N d-dimensional data points in a cluster: $\{x_i\}$ where $i = 1, 2, \ldots, N$, the centroid C and radius R of the cluster are defined as,

$$C = \frac{\sum_{i=1}^{N} x_i}{N} \tag{2.1}$$

$$R = (\frac{\sum_{i=1}^{N} ||x_i - C||^2}{N})^{\frac{1}{2}} \tag{2.2}$$

where R is the average distance from the member points to the centroid.

The concepts of *clustering feature* (CF) tree is at the core of the hierarchical micro-clustering algorithm which makes the clustering incremental without expensive computation. A CF is a triple which summarizes the information that a CF tree maintains for a cluster.

Definition 1 (Clustering Feature [59]). *Given n d-dimensional data points in a cluster: $\{x_i\}$ where i = 1, 2, ..., n, the CF vector of the cluster is defined as a triple: $CF = (n, LS, SS)$, where n is the number of data points in the cluster, LS is the linear sum of the n data points, i.e., $\sum_{i=1}^{n} x_i$, and SS is the square sum of the n data points, i.e., $\sum_{i=1}^{n} x_i^2$.*

Theorem 1 (CF Additivity Theorem [59]). *Assume that $CF_1 = (n_1, LS_1, SS_1)$ and $CF_2 = (n_2, LS_2, SS_2)$ are the CF vectors of two disjoint clusters. Then the CF vector of the cluster*

formed by merging the two disjoint clusters is:

$$CF_1 + CF_2 = (n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2) \tag{2.3}$$

Refer to [59] for the proof.

From the CF definition and additivity theorem, we know that the CF vectors of clusters can be stored and calculated incrementally and accurately as clusters are merged. The centroid C and the radius R of each cluster can be also computed from the CF of the cluster.

The CF is a summary of a set of data points which form a small cluster. Managing only this CF summary is efficient, saves spaces significantly, and is sufficient for calculating all the information for building the hierarchical micro-clusters which will facilitate computing an SVM boundary for a very large data set.

2.3.1.1 CF tree

A CF tree is a height-balanced tree with two parameters: branching factor b and threshold t. A CF tree of height $h = 3$ is shown at the right side of Figure 2.2. Each nonleaf node consists of at most b entries of the form $(CF_i, child_i)$, where (1) $i = 1, 2, \ldots, b$, (2) $child_i$ is a pointer to its i-th child node, and (3) CF_i is the CF of the subcluster represented by this child. A *leaf entry*, the entry in a leaf node, only has a CF without a child pointer. So, a leaf or a nonleaf node represents a cluster made up of all the subclusters represented by its entries. The threshold t is a constraint for the leaf entries to satisfy such that the radius of an entry in a leaf node has to be less than t.

The tree size is a function of t. The larger t is, the smaller the tree is. The branching factor b can be determined by a page size such that a leaf or a nonleaf node fits in a page.

This CF tree is a compact representation of the data set because each entry in a leaf node is not a single data point but a subcluster (which absorbs many data points with radius under a specific threshold t).

2.3.2 Algorithm Description

A CF tree is built up dynamically as new data objects are inserted. The ways that it inserts a data into the correct subcluster, merges leaf nodes, and manages nonleaf nodes are similar to those in a B+-tree, which can be sketched as follows:

1. **Identifying the appropriate leaf**: Starting from the root, it descends the CF tree by choosing the child node whose centroid is the closest.

2. **Modifying the leaf**: If the leaf entry can absorb the new data object without violating the threshold condition, update just the CF vector of the entry. Otherwise, add a new entry. If adding a new entry causes a node split, split the node by choosing the farthest pair of entries as seeds, and redistributing the remaining entries based on the closeness.

3. **Modifying the path to the leaf**: It updates the CF vectors of each nonleaf entry on the path to the leaf. Node split in the leaf causes an insertion of a new nonleaf entry into the parent node, and if the parent node is split, a new entry is inserted into the higher level node. Likewise, this occurs recursively to the root.

Due to the limited number of entries in a node, a highly skewed input could cause two subclusters that should have been in one cluster split across different nodes, and vice versa. These infrequent but undesirable anomalies are handled in the original BIRCH algorithm by further refinement with additional data scans. However, we do not perform this refinement because the infrequent and localized inaccuracy do not impact much the performance of CB-SVM.

2.3.2.1 Determination of threshold

The choice of the threshold t is crucial for building the tree in the right size which fits in the available memory because if t is too small, we run out of memory before all the data are scanned. The original BIRCH algorithm initially sets t very low, and iteratively increases t until the tree fits in the memory. Zhang et al. proved that rebuilding the tree with a larger t requires a re-scan of the data inserted in the tree so far and at most h extra pages of memory, where h is the height of the tree [59]. The heuristics for updating t_i is also provided in [59]. Due to the space limitation and to keep the focus of the paper, we skip the details of our implementation. In our experiments, we set the initial threshold t_1 intuitively based on the number of data points n, the dimensionality d, and the value range r_d of each dimension such that t_1 is proportional to $n * d * r_d$, and the tree of t_1 mostly fits in memory.

2.3.2.2 Outlier handling

After the construction of a CF tree, the leaf entries that contains far fewer data points than average are considered to be outliers. A low setting of outlier threshold can increase the classification performance of CB-SVM, especially when the data set is large compared to the size of dimensions and the boundary functions are simple (which is related to having a low VC dimension in machine learning theory). This is because the non-trivial amount of noise in the training data that may not be separable by a simple boundary function prevents the SVM boundary from converging in the quadratic programming. For this reason, we enable the outlier handling with a low threshold in our experiments in Section 2.5 because the type of data that we target is of a large number of data points with a relatively low dimensionality, and the boundary function is linear to the VC dimension $m + 1$ where m is the number of dimensions. See Section 2.5 for more details.

2.3.2.3 Analysis

A CF tree that fits in a memory can have at most $\frac{M}{P}$ nodes where M is the size of memory and P is the size of a node. The height h of a tree is $log_b \frac{M}{P}$, independent of the size of the data set. However, if we assume that memory is unbounded, and the number of the leaf entries is equal to the number of data points N due to a very small threshold t, then $h = log_b N$.

Insertion of a node into a tree requires the examination of b entries, and the cost per entry is proportional to the dimension d. Thus, the cost for inserting N data points is $O(N * d * b * h)$. In case of rebuilding the tree due to the poor estimation of t_1, additional re-insertions of the data already inserted has to be added in the cost. Then the cost becomes $O(k * N * d * b * h)$ where k is the number of the rebuildings. If we only consider the dependence of the size of the data set, the computation complexity of the algorithm is $O(N)$. Experiments from the original BIRCH algorithm have also shown the linear scalability of the algorithm with respect to the number of data points, and good quality of clustering of the data.

2.4 Clustering-Based SVM (CB-SVM)

In this section, we present the CB-SVM algorithm which trains a very large data set using the hierarchical micro-clusters (i.e., CF tree) to construct an accurate SVM boundary function.

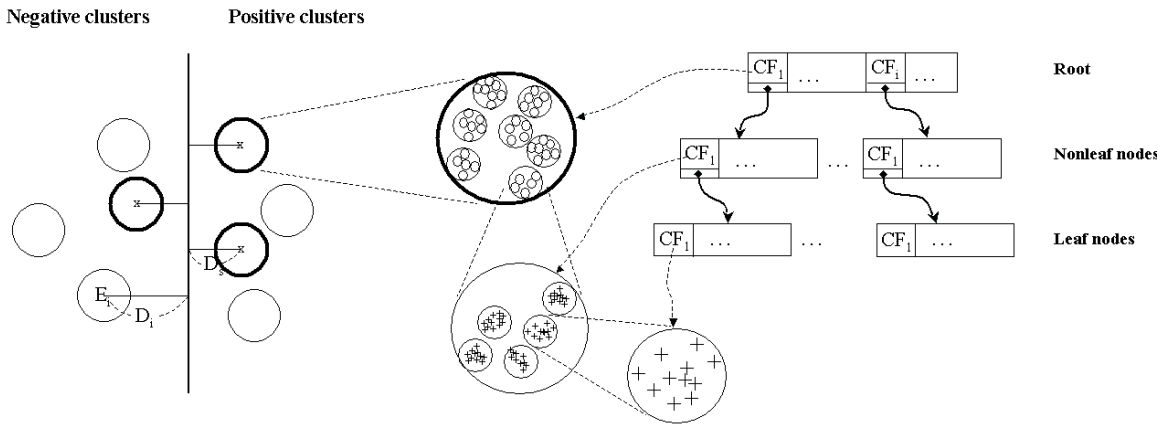


Figure 2.2 Example of the SVM boundary trained from the root entries of positive and negative trees.

The key idea of CB-SVM can be viewed being similar to that of selective sampling (or active learning), i.e., selecting the data which maximizes the benefit of learning. The selective sampling for SVM selects and accumulates the *low margin data* at each round that are close to the boundary in the feature space because the low margin data have higher chances to become the SVs of the boundary for the next round [49, 42]. Appreciating this idea, we decluster the entries near the boundary to get finer samples nearer to the boundary and coarser samples farther from the boundary. In this way, we induce the SVs, the description of the boundary, as fine as possible while keeping the total number of training data points as small as possible.

While selective sampling needs to scan the entire data set at each round to select the closest data point, CB-SVM runs based on the CF tree which can be constructed in a single scan of the entire data set and is carrying the statistical summaries that facilitates constructing an SVM boundary efficiently and effectively. The sketch of the CB-SVM algorithm is as follows.

1. Construct two CF trees from positive and negative data set independently.

2. Train an SVM boundary function using the centroids of the root entries. i.e., the entries in the root node, of the two CF trees. If the root node contains too few entries, train it using the entries of the nodes at the second levels of the trees.

3. Decluster the entries near the boundary into the next level, and the children entries declustered from the parent entries are accumulated into the training set with the non-declustered parent entries.

4. Construct another SVM from the centroids of the entries in the training set, and repeat from step 3 until nothing is accumulated.

The CF tree is a suitable base structure for CB-SVM to efficiently perform the selective declustering. The clustered data also provides better summaries for SVM than random samples because the random sampling is susceptible to a biased (or skewed) input, and thus it may generate undesirable outputs especially when the probability distributions of training and testing data are not similar, which is common in practice. (We discuss this in details in Section 2.5.)

2.4.1 CB-SVM Description

Let us first consider linearly separable cases.

Let *positive tree* T_p and *negative tree* T_n be the CF trees built from the positive data set and the negative data set respectively. We first train an SVM boundary function h from the *centroids of the root entries* of T_p and T_n. Note that each entry (or cluster) E_i contains the CF information from which we can efficiently compute the center point C_i and the radius R_i of the cluster. Figure 2.2 shows an example of the SVM boundary with the root clusters and the corresponding positive tree.

With the boundary function h and the root entries, we determine the *low margin clusters* that are close to the boundary and thus needs to be declustered into the finer level. Let *support clusters* be the clusters whose center points are the SVs of the boundary h, e.g., the bold circles in Figure 2.2. Let D_s be the distance from the boundary to the centroid of a support cluster, and let D_i be the distance from the boundary to the centroid of a cluster E_i. Then, we consider a cluster E_i which satisfies the following constraint as a *low margin cluster*.

$$D_i - R_i < D_s \tag{2.4}$$

where R_i is the radius of the cluster E_i.

The clusters that satisfy the constraint equation (2.4), have chances for their subclusters to be the support clusters of the boundary as illustrated in Figure 2.3, where five clusters initially satisfied the constraint equation (2.4), (three of them were the support clusters) and thus were declustered into finer levels, which results in the right figure of Figure 2.3. The subclusters whose parent clusters do not satisfy the constraint equation (2.4) would not be the support

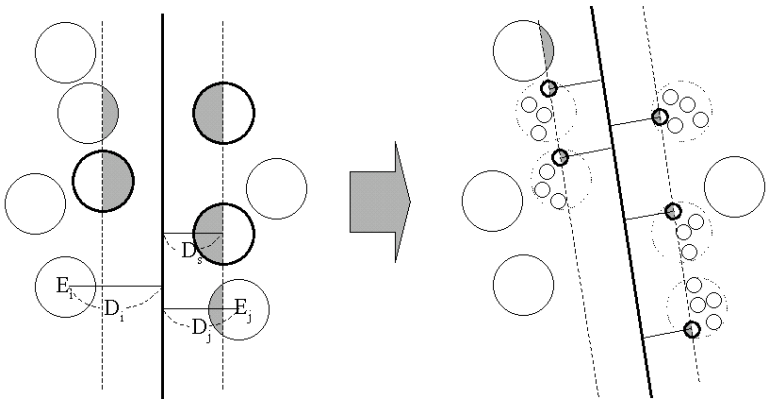


Figure 2.3 Declustering of the low margin clusters.

clusters of the boundary h because the surfaces of the parent clusters are farther than the SVs of the boundary. Thus we have the following remark.

Remark 1 (Hard Declustering Constraint). *Let R_i and D_i be the radius of a cluster E_i and the distance from the boundary to the centroid of the cluster respectively. Given a separable set of positive and negative clusters $E = \{E_i^+\} \cup \{E_i^-\}$ and the SVM boundary h of the set, the subclusters of E_i have the possibilities to be the support clusters of the boundary h only if $D_i - R_i < D_s$, where D_s is the distance from the boundary to the centroid of a support cluster.*

The example we illustrated was a linearly separable case with the hard constraints of SVM. In practice, the soft constraints are necessary to cope with noise in the training set. Using the soft constraints generates the SVs having different distances from the boundary. For the declustering condition with the soft constraints of SVM, we replace D_s with D_{ms}, *the maximum distance of all* D_s, which would include all the clusters whose subclusters have the possibilities to be the support clusters of the soft boundary h.

$$D_i - R_i < D_{ms} \tag{2.5}$$

The subclusters whose parent clusters do not satisfy constraint equation (2.5) would not be the support clusters of the soft boundary h because the surfaces of the parent clusters are farther than the most distant SV of the boundary.

Remark 2 (Soft Declustering Constraint). *For the soft constraints of SVM, the subclusters of E_i have the possibilities to be the support clusters of the boundary h only if $D_i - R_i < D_{ms}$,*

Input: - positive data set $\mathcal{P}$, negative data set $\mathcal{N}$
Output: - a boundary function h

Notation:
- $HC(\mathcal{S})$: a clustering algorithm that builds a hierarchical cluster tree T from a data set $\mathcal{S}$
- getRootEntries(T): return the root entries of a tree T
- getChildren($\mathcal{S}$): return the children entries of an entry set S
- getLowMargin(h, $\mathcal{S}$): return the low margin entries from a set S which are close to the boundary h (See Figure 2.5)

Algorithm:
1. $T_p = HC(\mathcal{P})$; $T_n = HC(\mathcal{N})$;
2. $\mathcal{S}$:= getRootEntries(T_p) $\cup$ getRootEntries(T_n);
3. Do loop
   3.1. h := SVM.train($\mathcal{S}$);
   3.2. $\mathcal{S}'$:= getLowMargin(h, $\mathcal{S}$);
   3.3. $\mathcal{S} := \mathcal{S} - \mathcal{S}'$;
   3.3. $\mathcal{S}'$:= getChildren($\mathcal{S}'$);
   3.4. Exit if $\mathcal{S}' = \emptyset$;
   3.5. $\mathcal{S} := \mathcal{S} \cup \mathcal{S}'$;
4. Return h;

Figure 2.4 CB-SVM

where D_{ms} is the maximum distance from the boundary to the centroids of all the support clusters.

Figures 2.4 and 2.5 describe the CB-SVM algorithm with the soft declustering constraint.

2.4.2 CB-SVM Analysis

As discussed in Section 2.3.2.3, building a CF tree costs $O(N)$ where N number of data points. (We disregard the number of dimensions in our analysis since it is linearly dependent.) Once the CF tree is built, the training time of CB-SVM becomes dependent on the number of entries instead of the number of data points.

Let us assume $t(SVM) = O(N^2)$ where $t(\Psi)$ is the training time of algorithm Ψ. The number of the leaf entries is at most b^h. Thus, $t(SVM)$ from the leaf entries becomes $O(b^{2h})$.

Let *support entries* be the SVs when the training data are entries of some nodes. Assume that r is the average rate of the support entries per the training entries. Namely, $r = s/b$ where

Input: - a boundary function h, a entry set $\mathcal{S}$
Output: - a set of the low margin entries $\mathcal{S}'$

Algorithm:
1. D_{SV} := getMaxDistanceOfSVs(h);
   // return the maximum distance of the support vectors
   from the boundary h
2. $\mathcal{S}'$:= getLowerMarginData(D_{SV}, $\mathcal{S}$);
   // return the data whose margin is smaller than D_{SV}
3. Return $\mathcal{S}'$;

Figure 2.5 getLowMargin(h, $\mathcal{S}$)

b is the number of training entries and s is the average number of support entries, e.g., $r = 0.01$ for $s = 10$ and $b = 1000$. Normally $s \ll b$ and $0 < r \ll 1$ for standard SVMs with large data sets.

Theorem 2 (Training Complexity of CB-SVM). *If the number of the leaf entries of a CF tree is equal to the number of training data points N, then CB-SVM trains asymptotically $1/r^{2h-2}$ times faster than standard SVMs given the CF tree, where r is the average rate of SVs and the height of the tree $h = log_b N$.*

Proof. If we approximate the number of iterations in CB-SVM $I \approx h$ (the height of CF tree), then the training complexity of CB-SVM given the CF tree is:

$$t(CBSVM) = \sum_{i=1}^{h} t_i(CBSVM) \tag{2.6}$$

where $t_i(CBSVM)$ is the training complexity of the i-th iteration of CB-SVM. The number of training data points N_i at the i-th iteration is:

$$\begin{aligned} N_i &= b - s + bs - s^2 + ... + bs^{i-2} - s^{i-1} + bs^{i-1} \\ &= (b-s)(1 + s + s^2 + ... + s^{i-2} + bs^{i-1}) \\ &= (b-s)\frac{s^{i-1} - 1}{s - 1} + bs^{i-1} \end{aligned}$$

where b is the number of data points in a node, and s is the number of the SVs among the data. If we assume $t(SVM) = O(N^2)$, by approximation of $s - 1 \approx s$,

$$\begin{aligned} t_i(CBSVM) &= O([bs^{i-2} + 1 - \frac{b}{s} - s^{i-1} + bs^{i-1}]^2) \\ &= O([bs^{i-1}]^2) \end{aligned}$$

If we accumulate the training time of all iterations,

$$\begin{aligned} t(CBSVM) &= O(\sum_{i=1}^{h}[bs^{i-1}]^2) = O(b^2 \sum_{h=0}^{h-1} s^{2i}) \\ &= O(b^2 \frac{s^{2h} - 1}{s^2 - 1}) \approx O(b^2 s^{2h-2}) \end{aligned}$$

If we replace s with br since $r = s/b$,

$$t(CBSVM) = O([b^h r^{h-1}]^2) = O(b^{2h} r^{2h-2})$$

Therefore, $t(CBSVM)$ trains asymptotically $1/r^{2h-2}$ times faster than $t(SVM)$ which is $O(b^{2h})$ for $N = b^h$. □

From Theorem 2, where $t(CBSVM) \approx O(b^2 s^{2h-2})$, we can see that the training time of CB-SVM is quadratically dependent on the number of support vectors (s^h) which is much less than that of the entire data (b^h). Normally $r \ll 1$, especially for very large data sets. So, the performance difference between CB-SVM and a standard SVM goes higher as the data set becomes larger.

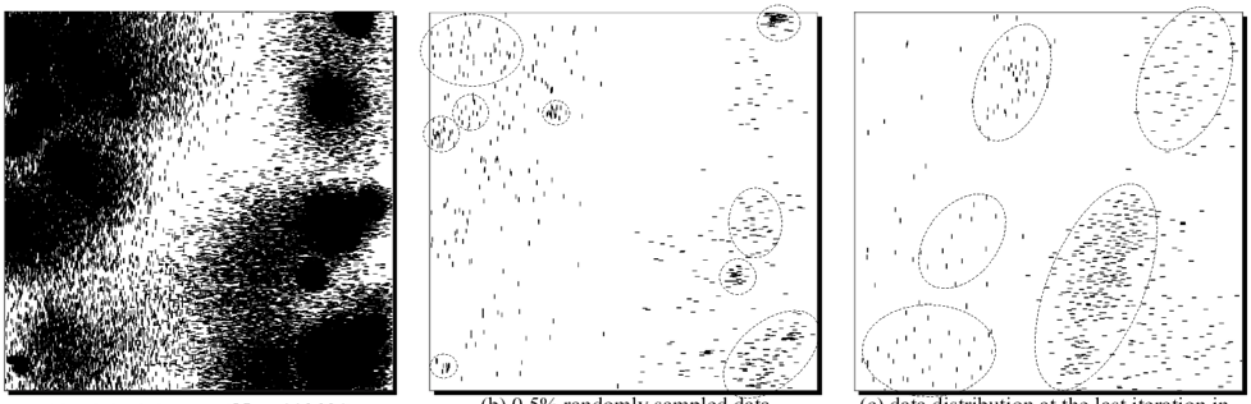
2.5 Experimental Evaluation

In this section, we provide empirical evidence of our analysis on CB-SVM using synthetic and real data sets, and discuss the results. All the experiments are conducted in a Pentium III 800Mhz machine with 906MB memory.

2.5.1 Synthetic data set

2.5.1.1 Data generator

To verify the performance of CB-SVM in a realistic environment while providing visualizing results, we perform binary classifications on two-dimensional data sets generated as follows.



(a) original data set ($N = 113601$)

(b) 0.5% randomly sampled data ($N = 603$)

(c) data distribution at the last iteration in CB-SVM ($N = 597$)

Figure 2.6 Synthetic data set in a two-dimensional space. '|': positive data; '−': negative data

1. We randomly created K clusters such that for each cluster, (1) the center point C is randomly chosen in the range $[C_l, C_h]$ for each dimension independently, (2) the radius R is randomly set in the range of $[R_l, R_h]$, and (3) the number of points n in each cluster is also randomly set in the range of $[n_l, n_h]$.

2. We labeled the clusters based on the X-axis value of each cluster such that cluster E_i is labeled as *positive* if $C_i^x < \theta - R_i$ and *negative* if $C_i^x > \theta + R_i$, where C_i^x is the X-axis value of the center C_i, and θ is the threshold value between C_l and C_h. We removed the clusters not assigned to either of positive or negative which lie across the threshold θ on X-axis. In this way, we drove the clusters to be linearly separable.

3. Once the characteristics of each cluster are determined, the data points for the cluster are generated according to a 2-d independent normal distribution whose mean is the center C, and whose standard deviation is the radius R. The class label of each data is inherited from the label of its parent cluster. Note that due to the properties of the normal distribution, the maximum distance between a point in the cluster and the center is unbounded. In other words, a point may be arbitrarily far from its belonging cluster. We refer to the points that belongs to cluster A but located farther than the surface of A as "outsiders". Due to the outsiders, the data set becomes not completely linearly separable, which is more realistic.

2.5.1.2 SVM parameter setting

We use the LIBSVM version 2.36[1] for SVM implementation and use ν-SVM with linear kernel. We enabled the shrinking heuristics for fast training [29]. ν-SVM has an advantage over standard SVMs: The parameter ν has a semantic meaning which denotes the upper bound of the noise rate and the lower bound of the SV rate in training data [12]. In our experiments, we set ν very low ($\nu = 0.01$ or $\nu = 0.1$) which usually performs very well when the size of data set is large and the noise is relatively small.

2.5.1.3 Results and discussion on a "large" data set

Figure 2.6(a) shows an example of the data set generated according to the parameters of Table 2.1. The data generated from the clusters on the left side and on the right side are positive ('|') and negative ('$-$') respectively.

Figure 2.6(b) shows 0.5% randomly sampled data from the original data set of Figure 2.6(a). Random sampling could hurt the SVM performance in the following ways:

- From Figure 2.6(b), we know that random sampling reflects the unstable data distribution of the original data set, which includes non-trivial amount of the unnecessary data points for training an SVM. The dashed ellipses on the figure indicating densely sampled areas. In practice, the areas around the boundary tends to be less dense because cluster centers which are likely very dense are unlikely to cross over the boundary of different classes. Thus the unnecessary data only increases the training time of SVM without contributing to the SVs of the boundary.

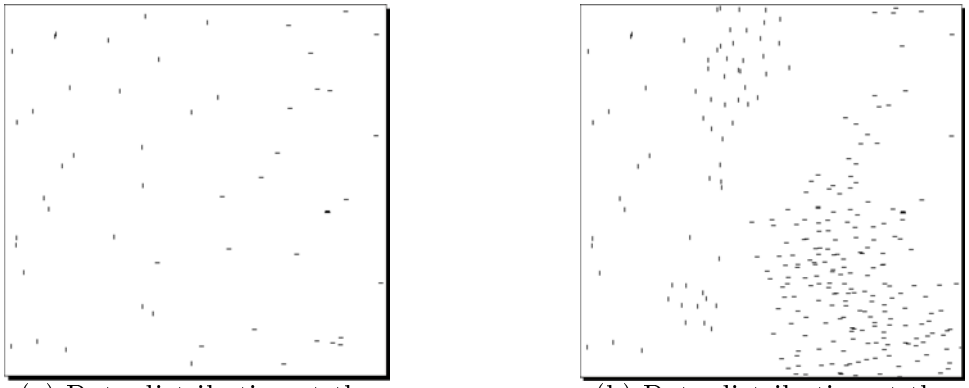
- Random sampling hurts more when the probability distributions of training and testing data are different because random sampling that only reflects the distribution of training data could miss significant regions of the testing data. (We show a real example of this in Section 2.5.2.)

Figure 2.6(c) shows the training data points at the last iteration in CB-SVM. We set $t_1 = 0.01$ and $b = 100$, and set the outlier threshold with the standard deviation. It generated a CF tree of $h = 3$, and CB-SVM iterated three times.

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm

Parameter	Values
Number of clusters K	50
Range of C $[C_l, C_h]$	[0.0, 1.0]
Range of R $[R_l, R_h]$	[0.0, 0.1]
Range of n $[n_l, n_h]$	[0, 10000]
θ	0.5

Table 2.1 Data generation parameters for Figure 2.6



(a) Data distribution at the first iteration ($N = 62$)

(b) Data distribution at the second iteration ($N = 308$)

Figure 2.7 Intermediate results of CB-SVM. '|': positive data; '−': negative data

Note that the training data points of CB-SVM are not the actual data but the summary of their clusters, so they tend not to have narrowly focused data points as it does in the random sampling. Also, the areas far from the boundary thus not likely to contribute to the SV will have very sparse data points because the clusters representing those areas would not be declustered in the process of CB-SVM.

Figure 2.7(a) and (b) show the intermediate data points that CB-SVM generated at the first and second iterations respectively. The data points in Figure 2.7(a) are the centroids of the root entries, which are very sparse. Figure 2.7(b) shows dense points around the boundary which are declustered into the second level of the CF tree. Finally, Figure 2.6(c) shows a better data distribution for SVM by declustering the support entries to the leaf level.

For fair evaluation, we generated a testing set using the same clusters and radiuses but different probability distributions by randomly re-assigning the number of points n for each cluster. We report the number of false predictions (# of false negative + # of false positive) on the testing data set because the data size is so big compared to the number of false prediction that the relative accuracy measure itself does not show much difference between them.

	Original	CB-SVM	0.5% samples
Number of data points	113601	597	603
SVM Training time (sec.)	160.792	0.003	0.003
Sampling time (sec.)	0.0	10.586	4.111
# of false predictions (# of FP, # of FN)	69 (49, 20)	86 (73, 13)	243 (220, 23)

Table 2.2 Performance results on synthetic data set (# of training data = 113,601, # of testing data = 107,072). FP:false positive; FN:false negative; Sampling time for CB-SVM: time for contructing the CF tree

Table 2.2 shows the performance results on the testing data set. *CB-SVM based on the clustering-based samples outperforms the standard SVM with the same number of random samples.* The "Number of data points" for CB-SVM in Table 2.2 denotes the number of training data points at the last iteration as shown in Figure 2.6(c). The "*training time*" for CB-SVM in the table indicates the SVM training time on that data, which is almost equal to that of 0.5% random samples since both generated similar number of data points. The "*sampling time*" for CB-SVM denotes the time to the construction of the 597 data points of Figure 2.6(c), which takes longer than the random sampling because it involves the construction of a CF tree. Note that the long construction time of the CF tree is partly caused by our non-optimized implementation of the hierarchical micro-clustering algorithm. However, as the data size grows, the random sample size that generates similar accuracies as that of CB-SVM also increases, for which the SVM training time ($O(N^2)$) becomes dominating over the "sampling time" for CB-SVM ($O(N)$), and thus the total training time of the SVM with random sampling ends up longer than that of CB-SVM. (See the next section.)

2.5.1.4 Results and discussion on a "very large" data set

We generated a much larger data set according to the parameters of Table 2.3 to verify the performance of CB-SVM compared to random sampling and SEL (selective sampling or active learning with SVM) and ASVM (active support vector machine) which is one of the most efficient linear SVM algorithms.[2] Table 2.4 shows the performance results of those on the "very large" data set. We could not run the standard SVM on more than 5% samples of the entire data set due to our limited resources (i.e., available memory and time). For ASVM [39], we only could run it with up to 10% of the data set. Running it with 20% of data exceeded our

[2]See http://www.cs.wisc.edu/dmi/asvm/ for the ASVM implementation.

Parameter	Values
Number of clusters K	100
Range of C $[C_l, C_h]$	[0.0, 1.0]
Range of R $[R_l, R_h]$	[0.0, 0.1]
Range of n $[n_l, n_h]$	[0, 1000000]
θ	0.5

Table 2.3 Data generation parameters for the very large data set

system resources and generated a system error. We discuss SEL and ASVM further in Section 2.7.

SEL and CB-SVM showed the error rates around $12 \sim 15\%$ lower than the random sampling and ASVM of the highest performance. The training time of CB-SVM in total (T-Time + S-Time) was much shorter than that of SEL, because SEL needs to scan the entire data set at each round to select the closest data point, which generates too much I/O cost to undergo as many rounds as it needs to get enough training data. SEL [42] showed the similar accuracy to CB-SVM since the basic idea is similar, which implies that *for large data sets, SVM performs better with a fine quality of samples than a large amount of random samples.* In this experiment, we ran the SEL with $\delta = 5$ (starting from one positive and one negative sample and adding five samples at each round), which gave fairly good results among others. (δ is commonly set below ten. If δ is too high, its performance converges slower, which ends up with a larger amount of training data to achieve the same accuracy, and if δ is too low, SEL may need to undergo too many rounds [42, 49].) It underwent 61 rounds resulting in 61 times of data scans to sample 307 training data, which took 56872.213 seconds in total for training.

2.5.2 Real data set

In this section, we experiment on a network intrusion detection data set from the UCI KDD archive which was used for the KDD Cup in 1999[3]. This data set consists of about five millions of training data and three hundred thousands of testing data. As previously noted, CB-SVM works better for very large data sets especially where random sampling hurts due to infrequent occurring important data or irregular patterns of the incoming data which cause different probability distributions of training and testing data. The network intrusion data set is a good example of such because the testing data is not from the same probability distribution

[3]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

S-Rate	# of data	# of errors	T-Time	S-Time
0.0001%	23	6425	0.000114	822.97
0.001%	226	2413	0.000972	825.40
0.01%	2333	1132	0.03	828.61
0.1%	23273	1012	6.287	835.87
1%	230380	1015	1192.793	838.92
5%	1151714	1020	20705.4	842.92
ASVM (1%)	230380	1172	26.4	838.92
ASVM (5%)	1152901	1037	183.78	829.81
ASVM (10%)	2303769	996	522.03	832.49
SEL	307	865	54872.213	
CB-SVM	2893	876	1.639	2528.213

Table 2.4 Performance results on the very large data set (# of training data = 23,066,169, # of testing data = 233,890). S-Rate: sampling rate; T-Time: training time; S-Time: sampling time; ASVM: active SVM; SEL: selective sampling

as the training data, and it also includes specific attack types not in the training data. (The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.) This is because they were collected in different times of periods, which makes the task more realistic. (Some intrusion experts believe that most novel attacks are variants of known attacks and the "signiture" of known attacks can be sufficient to catch novel variants.) Our experiments on this data set show that our method based on the clustering-based samples significantly outperforms the random sampling having the same number of samples.

2.5.2.1 Experiment setup

Each data object consists of 41 features (34 continuous features and 7 symbolic features). We normalized the continuous feature values into between zero and one by dividing them by their maximum values. We created one independent "zero-one" (predicate) feature for each value of the symbolic features such that "one" indicates the existence of the value. Our way of combining multi-variable features may not be the best way for SVM. Using more sophisticated techniques for pre-processing the features could improve the performance further.

We set $t_1 = 0.5$ for the CF tree because the total number of features in this data set becomes about 50 times larger than that in our synthetic data set and the range of each value is the same. The outlier threshold in this data set was tuned with a lower value because the outliers in the network intrusion data set could have valuable information. However, tuning the outlier threshold involves some heuristics depending on the type of data set and the type of boundary

S-Rate	# of data	# of errors	T-Time	S-Time
0.01%	515	25030	0.120689	502.59
0.1%	4917	25531	6.944	504.54
1%	49204	25700	604.54	509.19
5%	245364	25587	15827.3	524.31
ASVM (1%)	49204	25908	103.29	510.23
ASVM (5%)	245364	25787	633.51	508.72
ASVM (10%)	490566	28536	1460.65	514.19
SEL	747	21634	94192.213	
CB-SVM	4090	20938	7.639	4745.483

Table 2.5 Performance results on the network intrusion data set (# of training data = 4,898,431, # of testing data = 311,029). S-Rate: sampling rate; T-Time: training time; S-Time: sampling time; SEL: selective sampling

function. Further definition and justification on the heuristics for specific types of problems is a subsequent future work.

We used the same SVM implementation with the same parameters as in the experiments on the synthetic data sets. Linear kernel also showed good performance (over 90% accuracy) in this experiment, which implies the classification on this network intrusion data set is likely linearly separable.

2.5.2.2 Results

Our task is to distinguish normal connections from attacks. Table 2.5 shows the performance results of random sampling, ASVM, SEL, and CB-SVM. Running SVM with a larger amount of samples did not improve the performance much for the same reason as discussed in Section 2.5.1.4. SEL and CB-SVM also generated better results than the random sampling and ASVM, and the total training time of CB-SVM is much faster than that of SEL. (We run SEL with the same parameters as in Section 2.5.1.4.)

2.6 CB-SVM for Nonlinear Classification

In this section, we present a new declustering constraint for SVM nonlinear kernels, so that CB-SVM is also applicable to nonlinear classification by replacing function $getLowMargin(h, S)$ (Figure 2.5) in the CB-SVM algorithm of Figure 2.4 with a new $getLowMargin(h, S)$ function (Figure 2.8).

Input: - a boundary function h, a entry set $\mathcal{S}$
Output: - a set of the low margin entries $\mathcal{S}'$

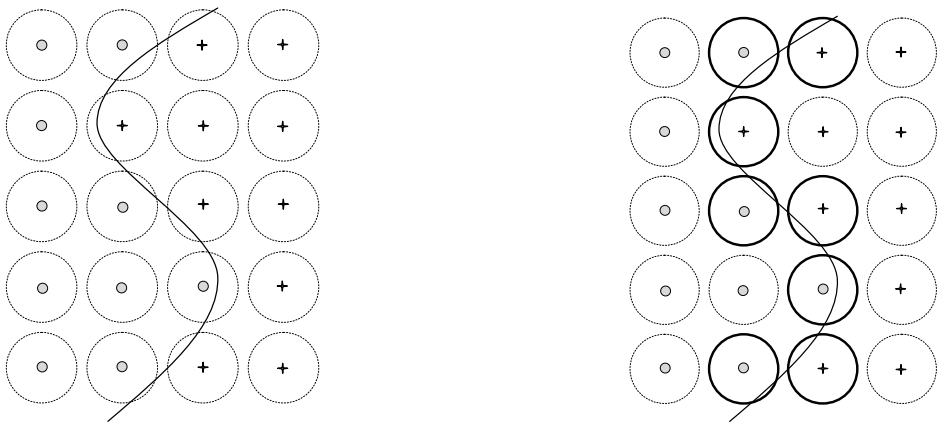
Algorithm:
1. $\mathcal{S}' = \emptyset$;
2. Do loop for s in $\mathcal{S}$;
   2.1. BS = createBorderSamples(s);
      // return artificial samples created around the border of cluster s
   2.2. if any of the samples in BS is misclassified by h, then $\mathcal{S}' = \mathcal{S}' \cup s$;
      // if any of the samples on the border is misclassified, the cluster needs to be declustered.
3. Return $\mathcal{S}'$;

Figure 2.8 getLowMargin(h, $\mathcal{S}$) for nonlinear kernels

The current declustering constraints of CB-SVM are not directly applicable to SVM nonlinear kernels because the statistical summaries (e.g., centroid and radius) computed in the input space cannot be used in the new feature space transformed by nonlinear kernels: distances in the new feature space are different from those in the input space. Thus, the declustering constraints $D_i - R_i < D_{ms}$ is not meaningful any more in the new feature space. Computing the radius in the nonlinear kernel space is possible by using the kernel trick for distances introduced in [45]. However, maintaining the centroid in the kernel space cannot be done incrementally. Thus, we here introduce an *artificial sampling technique* for the declustering constraint for nonlinear kernel. We first present the artificial sampling technique in the following section (Section 2.6.1) and verify the performance of our CB-SVM using a commonly-used nonlinear classification problem: the checkboard problem (Section 2.6.2).

2.6.1 Declustering Constraint for Nonlinear Kernels

An SVM classification function with a nonlinear kernel is represented as a nonlinear curve in input space while it is a linear function (i.e., a hyperplane) in feature space. To identify the clusters located close to the boundary, we generate artificial samples around the border of each cluster and test them using a current boundary function. If the samples generated from a cluster show inconsistent classification results for the current boundary function, the cluster are considered close to the boundary because the boundary passes through the cluster. We decluster only such clusters. To illustrate, consider Figure 2.9(a) which shows an example of



(a) (b)

Figure 2.9 Example of nonlinear boundary function in input space. '+': positive cluster center, 'o': negative cluster center

a nonlinear function in a two-dimensional input space. ('+' denotes a positive cluster center and 'o' denotes a negative cluster center.) SVM with a nonlinear kernel may draw a nonlinear function boundary shown in the figure, which correctly classifies two groups of the center points. The boundary passes through eight clusters which are shown as the bold circles in Figure 2.9(b). To determine whether a cluster is passed through by the boundary function, we generate artificial samples around the border of the cluster. If the boundary passes through the cluster, the artificial samples would show inconsistent classification results. That is, the samples generated from the eight clusters would show inconsistent classification results. To maximize the accuracy of this new declustering constraints while minimizing the number of artificial samples for efficiency, we create artificial samples only from the border of each cluster. Since we know the center C and radius R of each cluster, we can easily control generating the artificial samples.

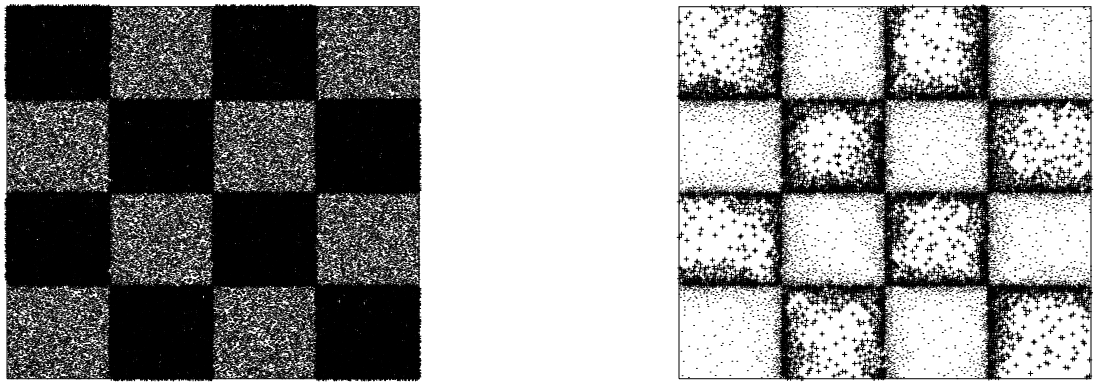
2.6.2 Experiment on Nonlinear Classification

To evaluate how effective the declustering constraint for nonlinear kernels is, we generated an artificial data set, checkboard (Figure 2.11(a)), which is commonly used to evaluate SVM nonlinear kernels especially the Gaussian kernels. Gaussian kernels are known to be the most

Input: - a cluster s: cluster center vector s_c, radius s_r
Output: - a set of artificial samples BS

Algorithm:
1. $BS = \emptyset$;
2. Do loop for each dimension i of s_c;
   2.1. Replace ith element e_i in s_c with $e_i + s_r$ and it to BS;
   2.2. Replace ith element e_i in s_c with $e_i - s_r$ and it to BS;
3. Return BS;

Figure 2.10 Example of *createBorderSamples(s)* which generates two samples for each dimension.



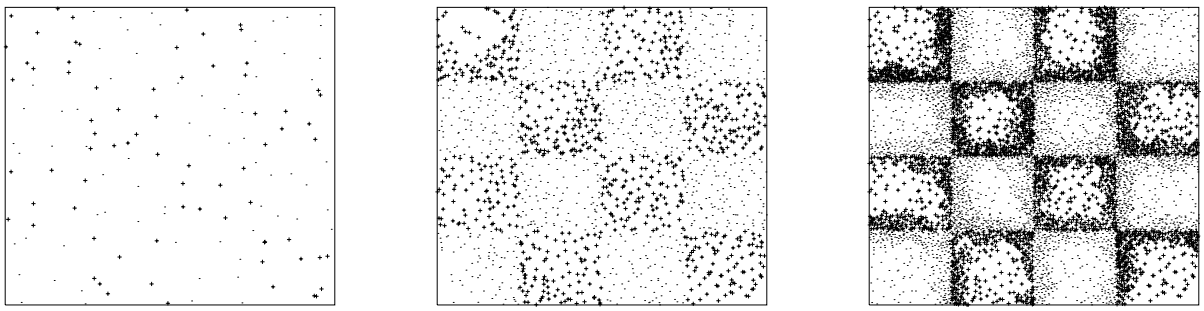
(a) # of data: 100,000 (b) # of data: 10,329

Figure 2.11 Experiment on a checkboard data set. '+': positive data; '·': negative data

complex and powerful[4]. We generated two samples for each dimension of a cluster as described in Figure 2.10.

Using the new declustering constraint for nonlinear kernels, Figure 2.12(a), (b), and (c) show intermediate results of CB-SVM on the checkboard data set. Figure 2.11(b) shows the data distribution after the 4-th iteration, which reduces the total number of training data ten times but generates the same quality of support vectors for the boundary. Thus, the total training time was significantly reduced while the quality of classification function is kept.

[4]In machine learning theory, the model complexity or the power of a classification function is often measured by VC-dimension, and SVMs with Gaussian kernels have infinite VC-dimensions [10], meaning that it is able to classify any possible partitions of a data set.



(a) after first iteration (b) after second iteration (c) after third iteration

Figure 2.12 Intermediate results of CB-SVM.

2.7 Related Work

Our work is in some aspects related to: (1) SVM fast implementations, (2) SVM approximations, (3) on-line SVM or incremental and decremental SVM for dynamic environments, (4) selective sampling (or active learning) for SVM, and (5) random sampling techniques for SVM.

Many algorithms and implementation techniques have been developed for training SVMs efficiently since the running time of the standard QP algorithms grows too fast. Most effective heuristics to speed up SVM training are to divide the original QP problem into small pieces, thereby reducing the size of each QP problem. Chunking, decomposition [29, 14], and sequential minimal optimization [41] are most well-known techniques. Our CB-SVM algorithm runs on top of these techniques to handle very large data sets by condensing further the training data into the statistical summaries of large data groups such that coarse summary is made for "unimportant" data and fine summary is made for "important" data.

SVM approximation has been attempted to improve the computational efficiency of SVM by altering the QP formulation to the extent that it keeps a similar semantic of the original SVM while it is faster to be solved by a QP solver [19, 1]. However, their new formulations are still not proven to be efficient and reliable enough to work with very large data sets.

Active SVM [39] is specially designed for linear SVM for relatively low dimensional spaces. Its target domains are similar to those of CB-SVM. Active SVM is currently known one of the most efficient linear SVM algorithms for such domains. However, as we showed in the experiments, it does not concern the limits of system resources and thus still has to experience the drawbacks of random sampling for very large data sets. Also, it becomes quadratically

slower as the number of dimensions increases since its training time is quadratically dependent on the number of dimensions.

On-line SVMs or incremental and decremental SVMs have been developed to handle dynamically incoming data efficiently [47, 11, 34]. In this senario that an SVM model is incrementally constructed and maintained, the newer data have a higher impact on the SVM model than older data. In other words, more recent data have a higher chance to be the SVs of the SVM model than older data. Thus, the analysis of archive data that treats all the data equally would generate undesirable outputs.

Selective sampling or active learning is to intelligently sample a small number of training data from the entire data set that maximizes the degree of learning, i.e., learning maximally with a minimum number of data points [23, 49, 42]. The core of the active learning technique is to select the data intelligently such that the degree of learning is maximized by the data. A common active learning paradigm iterates a training and testing process as follows: (1) construct a model by training an initially given data set, (2) test the entire data set using the model, (3) by analyzing the testing output, select the data (from the entire data set) that will maximize the degree of learning for the next round, (4) accumulate the data to the training data set, and train them to construct another model, and (5) repeat from steps (2) to (5) until the model becomes accurate enough.

The idea of selective sampling for SVM is to select the data close to the boundary in the feature space at each round because the data near the boundary have higher chances to be SVs in the next round, i.e., a higher chance to move the boundary further [49, 42]. They iterate until there exists no data nearer to the boundary than the SVs. However, an active learning system needs to scan the entire data set at every round to select the data, which generates too much I/O cost for very large data sets.

Some random sampling techniques [28] developed to reduce the training time of SVM for large data sets are also based on the same idea as the selective sampling that samples the data near the boundary with higher probabilities. They also need to scan the entire data set at each round when the samples are added in. Another method using random sampling [35] was developed for nonlinear SVM using the random sampling technique in the kernel trick.

2.8 Conclusions

This paper proposes a new method called CB-SVM (Clustering-Based SVM) that integrates a scalable clustering method with an SVM method and effectively runs SVM for very large data sets. Existing SVMs are not feasible to run such data sets due to their high complexity on the data size. CB-SVM applies a hierarchical micro-clustering algorithm that scans the entire data set only once to provide an SVM with high quality micro-clusters that carry the statistical summaries of the data such that the summaries maximize the benefit of learning the SVM. CB-SVM tries to generate the best SVM boundary for very large data sets given bounded memory based on the philosophy of hierarchical clustering where progressive deepening can be conducted when needed to find high quality boundaries for SVM. Our experiments on synthetic and real data sets show that CB-SVM is very scalable for very large data sets while generating high classification accuracy.

CHAPTER 3

Single-Class Classification via Support Vector Machines

3.1 Introduction

Single-Class Classification (SCC) seeks to distinguish one class of data from universal set of multiple classes. (e.g., distinguishing apples from fruits, identifying waterfall pictures from image databases, or classifying personal homepages from the Web) Throughout the paper, we call the target class *positive* and the complement set of samples *negative*.

In SCC problems, it is assumed that a reasonable sample of the negative data is hard to acquire. Since it is not natural to collect the "non-interesting" objects (i.e., negative data) to train the concept of the "interesting" objects (i.e., positive data), SCC problems are prevalent in real-world applications where positive and unlabeled data are widely available but negative data are hard or expensive to acquire [56, 36, 15]. For example, in text or Web page classification (e.g., personal homepage classification), collecting negative training data (e.g., a sample of "non-homepages") is delicate and arduous because manually collected negative data could be easily biased because of a person's unintentional prejudice, which could be detrimental to classification accuracy. For another example, consider the automatic diagnosis of diseases: unlabeled data are easy to collect (all patients in the database), and positive data are also readily available (the patients who have the disease). However, negative data can be expensive to acquire; not all patients in the database can be assumed to be negative if they have not been tested for the disease, and such tests can be expensive. Other applications can be found in pattern recognition, image retrieval, classification for data mining, rare class classification, etc. In this paper, we focus on SCC problem with positive and unlabeled data and without labeled negative data.

3.1.1 Previous Approaches for SCC

Traditional (semi-)supervised learning schemes are not suitable for SCC without labeled negative data because: (1) the portions of positive and negative spaces are seriously unbalanced (i.e., $Pr(P) << Pr(\overline{P})$) and hard to estimate, and (2) the absence of negative samples in the labeled data set makes unfair the initial parameters of the model and leads to unfair guesses for the unlabeled data.

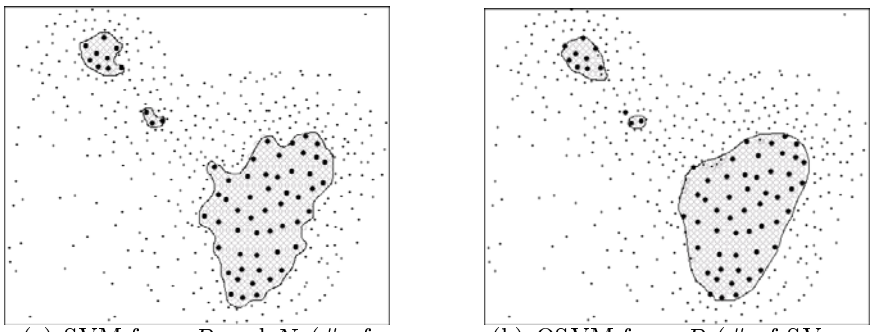
Active learning methods try to minimize the labeling labor to construct an accurate classification function by a different approach that involves an interactive process between the learning system and a user [49].

Valiant in 1984 [50] pioneered *learning theory from positive examples* based on rule learning. In 1998, F. Denis defined the Probably Approximately Correct (PAC) learning model for positive and unlabeled examples, and showed that k-DNF (Disjunctive Normal Form) is learnable from positive and unlabeled examples [16]. After that, attempts to learn using positive and unlabeled data have tried k-DNF or C4.5 [36, 15]. Such rule learning methods are simple and efficient for learning nominal features but are tricky to use for problems of continuous features, high dimensions, or sparse instance spaces.

Positive Example-Based Learning (PEBL) framework was proposed for Web page classification [56]. Their method is limited to the Web domain with binary features, and its training efficiency is poor as it uses SVM iteratively whose training time is already at least quadratic to the size of training data set. This problem becomes critical when the size of unlabeled data set is large.

Recently, a probabilistic method built upon the EM algorithm for the SCC problem, called *S-EM*, was proposed for the text domain [37]. The method has several fundamental limitations: the generative model assumption, the attribute independence assumption which results in linear separation, and the requirement of good estimation of prior probabilities. Our method does not require the prior probability of each class, and it can draw nonlinear boundaries using advanced SVM kernels.

The pattern recognition and verification fields have also explored various SCC methods, including neural network models [18, 22] and the SVMs [38, 8] (with increasing popularity). Some of these techniques tend to be domain specific: For instance, [8] uses SVM with only



(a) SVM from P and N (# of SVs = 229)

(b) OSVM from P (# of SVs = 15)

(c) Overfitting of OSVM (# of SVs = 21)

(d) Overfitting of OSVM (# of SVs = 52)

Figure 3.1 Boundaries of SVM and OSVM on a synthetic data set. *big dots: positive data, small dots: negative data*

positive examples for face detection– However, it relies on using the face features of other non-target classes as negative examples, and thus is not generally applicable to other domains.

For document classification, Manevitz [38] compared various SCC methods including neural network method, one-class SVM, nearest neighbor, naive Bayes, and Rocchio, and concluded that One-class SVM and neural network method were superior to all the other methods, and the two are comparable.

OSVM (One-Class SVM), based on the strong mathematical foundation of SVM, distinguishes one class of data from the rest of the feature space given only a positive data set [43, 48, 38]. OSVM draws a nonlinear boundary of the positive data set in the feature space using two parameters – ν (to control the noise in the training data) and γ (to control the "smoothness" of the boundary). They have the same advantages of SVM, such as efficient handling of high dimensional spaces and systematic nonlinear classification using advanced kernel functions. However, given no negative examples available, OSVM requires a much larger

amount of positive training data to induce an accurate class boundary especially in high dimensional spaces because its support vectors (SVs) of the boundary only comes from the positive data set and thus the relatively small number of SVs can hardly cover the major directions of the boundary in high dimensional spaces. Due to the SVs coming only from positive data, OSVM also tends to overfit and underfit easily. Tax proposed a sophisticated method which uses artifically generated unlabeled data to optimize the OSVM's parameters that "balance" between overfitting and underfitting [48]. However, their optimization method is infeasibly inefficient in high dimensional spaces, and even with the best parameter setting, its performance still lags far behind the SVM with negative data due to the shortage of SVs which makes "incomplete" the boundary description. Figure 3.1(a) and (b) show the boundaries of SVM trained from positives and negatives and OSVM trained from only positives on a synthetic data set in a two-dimensional space. (We used LIBSVM version 2.33[1] for SVM implementation.) In this low-dimensional space with "enough" data, the obstensibly "smooth" boundary of OSVM is not the result of the good generalization but instead is from the "incomplete" SVs due to not using the negatives for SVs, which will become much worse in high-dimensional spaces where more SVs around the boundary are needed to cover major directions in the high-dimensional spaces. When we increase the number of SVs in OSVM, it overfits rather than being more accurate as shown in Figure 3.1(c) and (d).

However, such OSVM boundary might be the best achievable one when only positive data are available. In this paper, under the assumption that unlabeled data are abundant and readily available, we present a systematic method that additionally uses the unlabeled data for support vectors to get more concrete boundary descriptions.

3.1.2 Contributions and Paper Layout

We present an SCC algorithm called *Mapping Convergence (MC)* that computes an accurate boundary of the target class from positive and unlabeled data (without labeled negative data) [55]. The basic idea of MC is to exploit the natural "gap" between positive and negative data by incrementally labeling negative data from the unlabeled data using the *margin maximization* property of SVM. We also present *Support Vector Mapping Convergence (SVMC)* which optimizes the MC algorithm for fast training. Our analyses show that MC and SVMC

[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvm

without labeled negative data significantly outperform other SCC methods. They generate as accurate boundaries as standard SVM with fully labeled data when the positive data is not very under-sampled and there exist gaps between positive and negative classes in the feature space. Our results also show that SVMC trains much faster than MC with very close accuracy.

In Section 3.2, we first discuss the optimal SCC boundary, which motivates our SCC method, *MC*, and present the MC algorithm which generates the boundary close to the optimum under certain conditions. We motivate and describe the SVMC algorithm in Section 3.3. In Section 3.4, we empirically verify our analysis of SVMC by extensive experiments on various domains of real data sets such as text classification, letter recognition, diagnosis of breast cancer, which shows the outstanding performance of SVMC in a wide spectrum of SCC problems (with linear or nonlinear separation, and low or high dimensions).

3.2 Mapping Convergence (MC) Algorithm

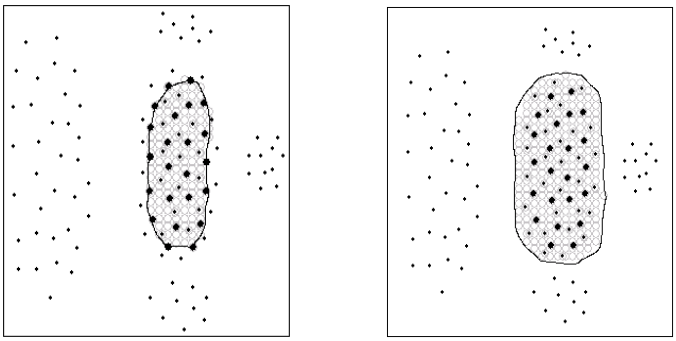
In this section, we present the basic idea of the Mapping-Convergence (MC) algorithm, which is the basis of the SVMC algorithm. For convenience of presentation, we use the following notations throughout this paper.

- $\mathcal{U}$ is the feature space for the universal set such that $\mathcal{U} \subseteq \Re^m$ where m is the number of dimensions.
- U (unlabeled data set) is an iid (independently and identically distributed) sample of the universal set.
- x is a data instance such that $x \in \mathcal{U}$.
- $\mathcal{P}$ is a subspace for positive class within $\mathcal{U}$, from which positive data set P is sampled.

For an example of Web page classification, the universal set is the entire Web, U is a sample of the Web, P is a collection of Web pages of interest, and $x \in \Re^m$ is an instance of Web page.

3.2.1 Motivation

In machine learning theory, the *optimal* class boundary function (or hypothesis) $h(x)$, given a limited number of training data $\{(x, l)\}$ (l is the label of x), is the one that gives the best



(a) OSVM from P (# of SVs = 7)

(b) SVMC from P and U (# of SVs = 75)

Figure 3.2 Synthetic data set simulating a real situation. *P: big dots, U: all dots (big and small dots)*

generalization performance which is the performance on "unseen" examples rather than on the training data. The performance on the training data is not regarded as a good evaluation measure for a hypothesis because the hypothesis may end up *overfitting* when it tries too hard to fit the training data. When a problem is easy (to classify) and the boundary function is more complicated than it needs to be, the boundary is likely overfitting. When a problem is hard and the classifier is not powerful enough, the boundary is likely underfitting. SVM is an excellent example of supervised learning that tries to maximize the generalization by maximizing the *margin* and also supports nonlinear separation using advanced kernels, by which SVM tries to avoid overfitting and underfitting [51, 13, 10].

To find the optimal SCC classifier without labeled negative data, we also needs to maximize the generalization performance so that we do not overfit nor underfit the training data. The optimal SCC boundary is much harder to find than one with complete labels for all the training data. To illustrate an example of desirable SCC boundary when no labeled negative data is available, consider the synthetic data set (Figure 3.2) which simulates a real situation where (1) $\mathcal{U}$ is the universal set consisting of multiple groups of data, (2) the positive class $\mathcal{P}$ is one of them (say the data group in the center), and (3) the positive data set P (represented by the big dots) is a sample from $\mathcal{P}$. OSVM draws a very conservative tight boundary around P as shown in Figure 3.2(a), clearly overfitting the data due to its inability of using any knowledge about the distribution of U. Intuitively, the desirable boundary should be located between $\mathcal{P}$ and U outside $\mathcal{P}$. The MC algorithm exploits U systematically to obtain a reasonable set of negative data, which allows us to reach such a desirable boundary shown in Figure 3.2(b).

3.2.2 Strong Negative

Here we introduce the notion of "strong negative", which is key to the MC algorithm.

Let $h(x)$ be the boundary function of the positive class in $\mathcal{U}$, which outputs the distance from the boundary to the instance x in $\mathcal{U}$ such that

$$\begin{aligned} h(x) > 0 \qquad & if\ x\ is\ a\ positive\ instance, \\ h(x) < 0 \qquad & if\ x\ is\ a\ negative\ instance, \\ |h(x)| > |h(x')| \qquad & if\ x\ is\ located\ farther\ than\ x' \\ & from\ the\ boundary\ in\ \mathcal{U}. \end{aligned}$$

Definition 2 (Strong negative). *For two negative instances x and x' such that $h(x) < 0$ and $h(x') < 0$, if $|h(x)| > |h(x')|$, then x is **stronger** than x'.*

Example 1. *Consider a resume page classification function $h(x)$ from the Web ($\mathcal{U}$). Suppose there are two negative data objects x and x' (non-resume pages) in $\mathcal{U}$ such that $h(x) < 0$ and $h(x') < 0$: x is "how to write a resume" page, and x' is "how to write an article" page. In $\mathcal{U}$, x' is considered a stronger negative (or more distant from the boundary of the resume class) because x has more features relevant to the resume class (e.g., the word "resume" in text) though it is not a true resume page.*

3.2.3 MC Algorithm

The MC algorithm is composed of two stages: the *mapping stage* and the *convergence stage*. In the mapping stage, the algorithm uses a weak classifier Ψ_1 (e.g., Rocchio or OSVM), to draw an initial approximation of "strong negatives" – the negative data located far away from the boundary of the positive class in $\mathcal{U}$ (steps 1 and 2 in Figure 3.3). Based on this initial approximation, the convergence stage runs iteratively using a second base classifier Ψ_2 (e.g., SVM), to maximize the margin in order to make a progressively better approximation of negative data (steps 3 through 5 in Figure 3.3). As a result, the class boundary eventually converges to the boundary around the positive data set in the feature space.

The MC algorithm is described in Figure 3.3. To illustrate the MC process, consider an example of data distribution in a one-dimensional feature space in Figure 3.4. U is composed of eight data clusters. The fifth one from left is positive and the rest are negative. If U

Input: - positive data set P, unlabeled data set U
Output: - a boundary function h_i

Ψ_1: an algorithm identifying "strong negatives" from U
e.g., Rocchio or OSVM
Ψ_2: a supervised learning algorithm that maximizes the margin
e.g., SVM

Algorithm:
1. Use Ψ_1 to construct a classifier h_0 from P and U which classifies only "strong negatives" as negative and the others as positive
2. Classify U by h_0
   * $\hat{N}_0$:= examples from U classified as negative by h_0
   * $\hat{P}_0$:= examples from U classified as positive by h_0
3. Set $N := \emptyset$ and $i := 0$
4. Do loop
   4.1. $N := N \cup \hat{N}_i$
   4.2. Use Ψ_2 to construct h_{i+1} from P and N
   4.3. Classify $\hat{P}_i$ by h_{i+1}
      * $\hat{N}_{i+1}$:= examples from $\hat{P}_i$ classified as negative by h_{i+1}
      * $\hat{P}_{i+1}$:= examples from $\hat{P}_i$ classified as positive by h_{i+1}
   4.4. $i := i + 1$
   4.5. Repeat until $\hat{N}_i = \emptyset$
5. return h_i

Figure 3.3 MC algorithm

is completely labeled, a margin-maximization algorithm such as SVM trained from U would generate the optimal boundary (b_d, b'_d), where b_d maximizes the margin – the gap between the two points g_n and g_f – between positive and negative clusters. Unfortunately, under the common scenarios of SCC, the only labeled data are the dark center within the positive cluster, and the rest are unlabeled.

Figure 3.5(a) illustrates the MC process given the labeled positive data P (i.e., the dark center) with the unlabeled U. Let the strong negatives that algorithm Ψ_1 identifies be the data to the left and right sides of b_1 and b'_1 respectively. They are located far from the positive data cluster in the feature space. We will justify the validity of algorithm Ψ_1 later in this section. At each iteration of Step 4 in Figure 3.3, algorithm Ψ_2 incorporates the data within D_i into N

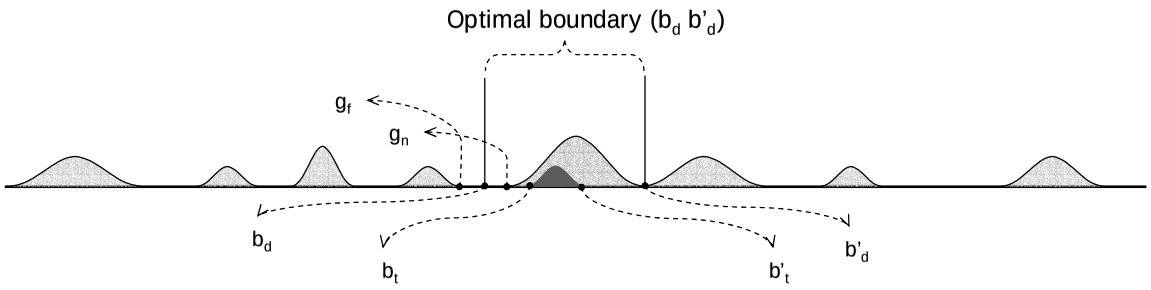


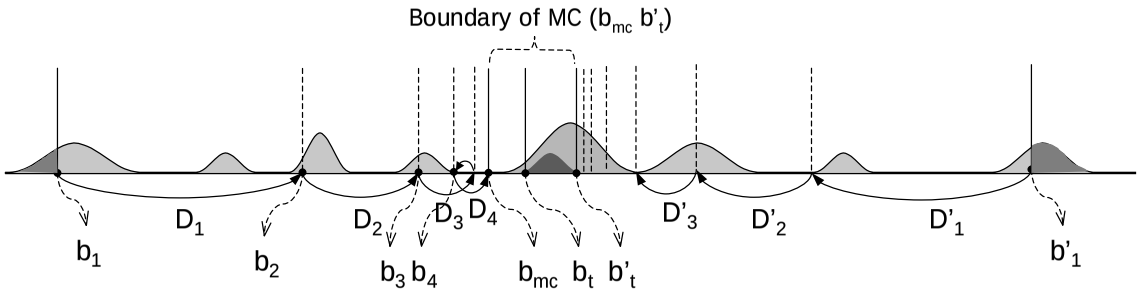
Figure 3.4 Example of data distribution in one dimensional feature space. The fifth data cluster from left is positive and the rest are negative.

as shown in Figure 3.5(a), where i is the number of iterations and b_i is the starting point of ith iteration in the feature space. Since Ψ_2 maximizes the margin at each iteration, D_1 has the half margin of that between b_1 and b_t, and D_{i+1} will have the half margin of D_i.

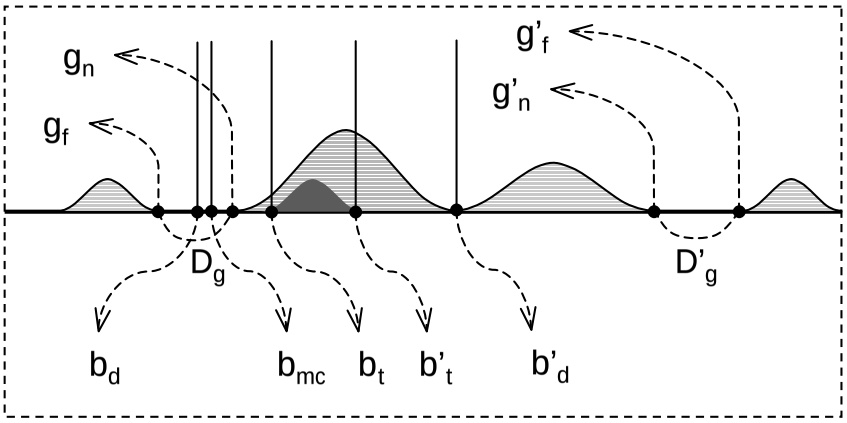
If there exists no data around the boundary b_{i+1} after ith iteration, b_{i+1} will retract to a point where there exists data within D_i, because the data within D_i merged into N will be the nearest negative data points of the next boundary, and Ψ_2 maximized the margin between the nearest data points (i.e., support vectors). For instance, in Figure 3.5(a), b_4 retracts a little from the boundary after the 3rd iteration.

If there exists no data at all within D_i, the convergence will stop because nothing will be added to N and thus more iterations after that will not affect the boundary. For instance, in Figure 3.5(a), b_{mc} will be the final boundary of MC *to the left side* because there exists no data within D_4 and thus nothing will be incorporated into N from that point. As we see from Step 4.5 of Figure 3.3, the MC algorithm will stop the convergence when nothing is incorporated into N. The final boundary of MC *to the right side* in the figure tightly fits around P (i.e., b'_t) because there is no gap between the positive and negative clusters and thus the convergence will not stop until MC includes every unlabeled data beyond P into N.

From the above example, we see that MC will stop the convergence and locate the boundary close to the optimum if there exists a large gap between positive and negative data clusters. (e.g., b_{mc} is close to b_d in Figure 3.5(b).) b_d equally divides the margin between g_f (i.e., farther point of the gap) and g_n (nearer point of the gap) and b_{mc} does the same between g_f and b_t. However, if there is no wide gap between those (e.g., the right side of Figure 3.5(a)) or the



(a) MC process.



(b) Zoomed-in center.

Figure 3.5 Example of the MC algorithm in the one dimensional feature space.

positive labeled data is very few, the MC boundary will "over-iterate" and end up converging into the tight boundary (e.g., b'_t in Figure 3.5(a)).

How wide gaps or how many positive labeled data are needed to avoid the "over-iteration" problem? To provide an intuitive answer, we assume that the feature space has only one dimension and negative data exist only to one side (e.g., the left side of Figure 3.5(a)). We denote $|x - y|$ for the margin between two points x and y. Let D_g be the gap between g_n and g_f where g_n and g_f are the two ending points of the gap respectively near and far from the positive class as shown in Figure 3.5(b). Then we have the following lemma.

Lemma 1. *If MC stop converging at ith iteration, there must exists a gap D_g such that $|g_f - g_n| > |g_n - b_t|$ and $b_i = g_f$, where b_i is the starting point of ith iteration.*

Proof. First, if MC stops the convergence at ith iteration, b_i will be the starting point of the gap (i.e., g_f), because if g_f is farther than b_i from b_t, b_i retracts to g_f, and if there exist data on b_i, MC will not stop the convergence and continue to accumulate the data to N such that $b_{i+1} = g_f$. Second, MC includes the half margin of $|b_i - b_t|$ from b_i at ith iteration since Ψ_2 maximizes the margin. If MC stops the convergence at ith iteration, there must exists a gap at least from b_i to the half point to b_t, i.e., $|g_f - g_n| > \frac{|g_f - b_t|}{2}$ since $b_i = g_f$. Then, $|g_f - g_n| > \frac{|g_f - b_t|}{2} = \frac{|g_f - g_n| + |g_n - b_t|}{2}$, Thus $|g_f - g_n| > |g_n - b_t|$ □

Theorem 3. *MC locates the boundary within the first gap D_g it confronts such that $|g_f - g_n| > |g_n - b_t|$ during the iterations.*

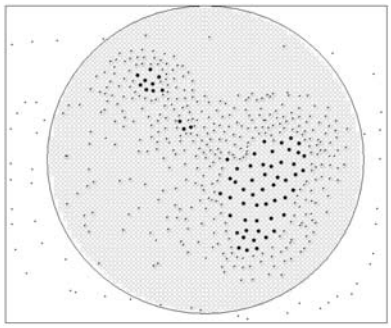
Proof. From Lemma 1, we know that if MC do not see a gap D_g such that $|g_f - g_n| > |g_n - b_t|$ during the iterations, it will not stop the convergence. Let us assume that MC confronts a gap of D_g such that $|g_f - g_n| > |g_n - b_t|$ at the ith iteration. Then, MC tries to include the margin of $\frac{|b_i - b_t|}{2}$ from b_i. If b_i is farther from b_t than g_f, $b_{i+1} = g_f$. If g_f is farther from b_t than b_i, $b_i = g_f$. Since $|g_f - g_n| > |g_n - b_t|$, $\frac{|g_f - b_t|}{2} = \frac{|g_f - g_n| + |g_n - b_t|}{2} < |g_f - g_n|$ Thus, nothing will be added to N at ith iteration, and the iteration will stop then. □

Note that our algorithm analyses so far assume that data is noiseless and Ψ_2 is hard-margin SVM. In practice, we use a soft-margin SVM for Ψ_2 to deal with noise in the training data. With soft-margin SVMs, D_{i+1} may not exactly have the half margin of D_i. A small soft-margin parameter (e.g., $\nu = 0.1$ or 0.01 in ν-SVM) performs well in SCC as our experiments in Section 3.4 also verify, because in SCC, P is likely to be a carefully collected subset of positive data and thus likely to be noise-free.

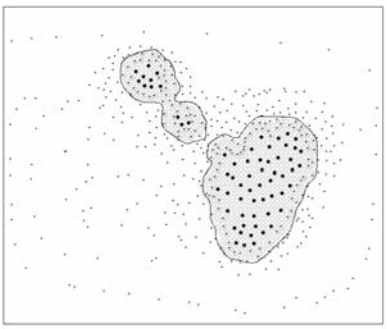
Although we analyze our algorithm in a one-dimensional feature space for convenience of explanation, our analyses can be generalized into the cases of high dimensional spaces if Ψ_2 maximizes the margin on each dimension in the feature space. We will discuss the requirement of Ψ_1 and Ψ_2 at the end of this section.

From Theorem 3, we can deduce the following observations:

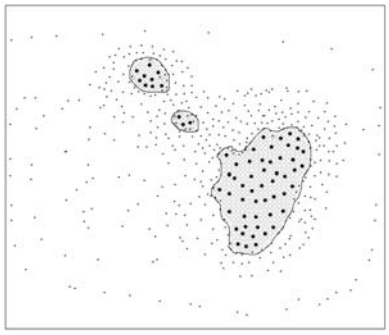
1. MC is not likely to locate the boundary exorbitantly far from the boundary of P unless U is extremely sparse, because a larger gap ($|g_f - g_n|$) is needed to stop the convergence as g_n (i.e., the nearer ending point of the gap) is becoming farther from b_t (or $|g_n - b_t|$ increases).



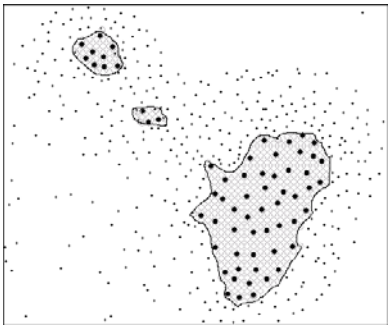
(a) Boundary after the mapping stage



(b) Boundary after the first iteration



(c) Boundary after the second iteration



(d) Final boundary

Figure 3.6 Intermediate results of SVMC

2. The more P is under-sampled, the larger the gaps between positive and negative classes are needed to avoid the "over-iteration" problem, because $|g_n - b_t|$ would increase if P is under-sampled.

The above observations imply that MC is likely to generate the boundary close to the optimum when P is not seriously under-sampled and wide gaps exist between P and N in the feature space. The data sets in practice are likely to have natural gaps between different classes than within a class in the feature space. From the first observation, gaps far from P in the feature space are not likely to be influential to MC. By exploiting such gaps, MC significantly outperforms other SCC methods when P is not too much under-sampled. Our experiments on common data sets in Section 3.4 show consistent results.

Note that the MC algorithm is quite general, as long as the component algorithms Ψ_1 and Ψ_2 satisfy the following requirements:

1. Ψ_1 *identifies strong negatives without including false negatives.*

If Ψ_1 includes false negatives, they are likely to stay as the false negatives during the iterations which will deteriorate the boundary accuracy. We can use any reasonable classifier, such as Rocchio or OSVM, and adjust the threshold so that it makes near 100% recall by sacrificing precision. We use OSVM for Ψ_1 in our research, and set the bias b very low to achieve a high recall. The precision of Ψ_1 does not affect the accuracy of the final boundary as long as it approximates a certain amount of negative data because the final boundary will be determined by Ψ_2. Figure 3.6 shows an example of the boundary after each iteration of SVMC. The mapping stage only identifies very strong negatives by covering a wide area around the positives (Figure 3.6(a)). Although the precision quality of mapping is poor, the boundary at each iteration converges (Figures 3.6(b) and (c)), and the final boundary is very close to the ideal boundary drawn by SVM on P and *true* N (Figure 3.1(a) and 3.6(d)). Our experiments in Section 3.4 also show that the final boundary becomes very accurate although the initial boundary of the mapping stage is very rough by the "loose" setting of the threshold of Ψ_1.

2. Ψ_2 *must maximize margin.*

   SVM and Boosting are currently the most popular supervised learning algorithms that maximize margin. We use SVM for Ψ_2 in our research. With a strong mathematical foundation, SVM automatically finds the optimal boundary without a validation process and without many parameters to tune. The small numbers of theoretically motivated parameters also work well with an intuitive setting. In practice, the soft constraint of SVM is necessary to cope with noise or outliers; however, in the SCC problem domains, P is unlikely to have a lot of noise since it is usually carefully labeled by users. Thus we can use a very low value for the soft margin parameter ν. In our experiments, a low setting (i.e., $\nu = 0.01$ or 0.1) of ν (the parameter to control the rate of noise in the training data) performed well for this reason. (When $\nu = 0$, it becomes the hard margin.) (We used ν-SVM for the semantically meaningful parameter [44, 12].)

3.3 Support Vector Mapping Convergence (SVMC)

In this section, we present *Support Vector Mapping Convergence (SVMC)* which optimizes the MC algorithm for fast training.

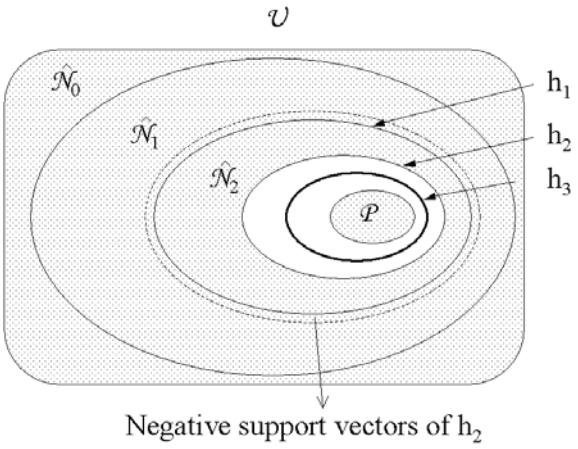


Figure 3.7 Minimally required negative data for h_3

3.3.1 Motivation

The classification time of the final boundary of MC with $\Psi_2 = SVM$ is equal to that of SVM because the final boundary is a boundary function of Ψ_2. However, the training time of MC can be very long if $|U|$ is very large because the training time of SVM highly depends on the size of data set n ($\approx |U|$), and MC runs iteratively. Indeed, $t_{MC} = O(|U|^2 * log|U|)$ assuming *the number of iterations* $\approx log|U|$ and $t_{SVM} = O(|U|^2)$ where t_Ψ is the training time of a classifier Ψ. (t_{SVM} is known to be at least quadratic to n and linear to the number of dimensions; More discussion on the complexity of SVM can be found in [12].) However, decreasing the sampling density of U to reduce the training time could hurt the accuracy of the final boundary because the density of U will directly affect the quality of the SVs of the final boundary. Fortunately, it is possible to prevent the training time from increasing dramatically as the sample size grows, and this leads to the SVMC algorithm, which implements the MC algorithm with the training time being *independent* of the number of iterations and asymptotically equal to that of a SVM.

3.3.2 SVMC Algorithm

The basic idea of SVMC is to use a minimally required data set at each iteration such that the data set does not degrade the accuracy of the boundary. In this way, it saves the training time of each SVM maximally. To illustrate how SVMC achieves this, consider the point of starting the third iteration (when $i = 2$) in MC (Step 4.1 in Figure 3.3.) illustrated in Figure 3.7. After we merge $\hat{N}_2$ into N, in order to construct h_3, we may not need all the data from N

since the data far from h_3 will not contribute to the SVs. The set of negative SVs of h_2 is the representative data set for $\hat{N}_0$ and $\hat{N}_1$, so we only keep the negative SVs of h_2 and the newly induced data set $\hat{N}_2$ to support the negative side of h_3.

Claim 1 (Minimally required negative data). *$\forall i \geq 1$, the minimally required negative data at $(i+1)$th iteration is $\hat{N}_i \cup$ negative SVs of h_i, which, along with P, can generate a h_{i+1} that is as accurate as one constructed from $\cup_{j=0}^{i} \hat{N}_j$ and P.*

Rationale. The negative SVs of h_{i+1} will be from $\hat{N}_i$ and the negative SVs of h_i because $\hat{N}_i$ is the closest data set to h_{i+1} and because the directions not supported by $\hat{N}_i$ in the feature space will be supported by the negative SVs of h_i, which are the representing data set for $\cup_{j=0}^{i-1} \hat{N}_j$. However, if any of the negative SVs of h_i is excluded in constructing h_{i+1}, h_{i+1} might suffer because the negative SVs of h_i need to support the direction that $\hat{N}_i$ does not support in the feature space. Thus, $\hat{N}_i \cup$ *negative support vectors of* h_i are the minimally required negative data set at $(i+1)$th iteration.

However, Claim 1 only applies for the hard margin cases (i.e., when $\nu = 0$). In practice, due to the soft margin, the negative SVs of h_i may not totally represent $\cup_{j=0}^{i-1} \hat{N}_j$ especially in high dimensional spaces. In our experiments, MC performed a little better than SVMC for this reason.

For the minimally required data set for the *positive* side, we can never be sure whether we can exclude any data object from P at each iteration, because with the negative SVs different at each iteration, it is hard to predict which positive data will not be SVs independent of the number of iterations.

Figure 3.8 shows the SVMC algorithm which uses the minimally required data set in each iteration. Surprisingly, adding only Step 4.5 to the original MC algorithm completes SVMC.

Theorem 4 (Training time of SVMC). *Suppose $t_{SVM} = O(n^2)$, and $|\hat{N}_{i+1}| \approx \frac{|\hat{N}_i|}{k}$, $\forall i \geq 1$ and $k > 1$. Then, $t_{SVMC} = O(n^2)$.*

Proof. For simplicity of the proof, we approximate each value as follows.

$$\begin{aligned} n = |P| + |\cup_{i=0}^{I} \hat{N}_i| &\approx |U| \\ |\hat{N}_0 \cup \hat{N}_1| &\approx \frac{|U|}{2} \end{aligned}$$

Input: - positive data set P, unlabeled data set U
Output: - a boundary function h_i

Ψ_1: Rocchio or OSVM
Ψ_2: SVM

Algorithm:
1. Use Ψ_1 to construct a classifier h_0 from P and U which classifies only "strong negatives" as negative and the others as positive
2. Classify U by h_0
   * $\hat{N}_0$:= examples from U classified as negative by h_0
   * $\hat{P}_0$:= examples from U classified as positive by h_0
3. Set $N := \emptyset$ and $i := 0$
4. Do loop
   4.1. $N := N \cup \hat{N}_i$
   4.2. Use Ψ_2 to construct h_{i+1} from P and N
   4.3. Classify $\hat{P}_i$ by h_{i+1}
      * $\hat{N}_{i+1}$:= examples from $\hat{P}_i$ classified as negative by h_{i+1}
      * $\hat{P}_{i+1}$:= examples from $\hat{P}_i$ classified as positive by h_{i+1}
   4.4. $i := i + 1$
   4.5. Reset N with negative SVs of Ψ_2
   4.6. Repeat until $\hat{N}_i = \emptyset$
5. return h_i

Figure 3.8 SVMC algorithm

$$
\begin{aligned}
t_{SVMC} &= (\frac{|U|}{2})^2 + (\frac{|U|}{2k})^2 + ... + (\frac{|U|}{2k^{I-1}})^2 \\
&= \sum_{i=1}^{I} (\frac{|U|}{2k^{i-1}})^2 = (\frac{|U|}{2})^2 \sum_{i=1}^{I} (\frac{1}{k^{i-1}})^2 = (\frac{|U|}{2})^2 (\frac{1 - 1/k^n}{1 - 1/k}) \\
&= (\frac{|U|}{2})^2 (\frac{k}{k-1} - \frac{1}{k^{n-1}(k-1)}) \approx (\frac{|U|}{2})^2 \\
&= O(n^2)
\end{aligned}
$$

□

Theorem 4 states that the training complexity of SVMC is asymptotically equal to that of SVM. Our experiments in Section 3.4 also show that SVMC trains much faster than MC with very close accuracy. As we show in Figure 3.6, under the certain conditions, the final boundary

of SVMC becomes very close to the ideal boundary drawn by SVM with P and true N (Figure 3.1(a) and 3.6(d)).

3.4 Experimental Evaluation

In this section, we provide the empirical evaluations on MC and SVMC by performing extensive experiments on various domains of real data sets – text classification, letter recognition, and diagnosis of breast cancer – which show the outstanding performance of SVMC in a wide spectrum of SCC problems (with linear or nonlinear separation, and low or high dimensions). Our evaluations show that MC and SVMC without labeled negative data significantly outperform other SCC methods and in most cases generate as accurate boundaries as standard SVM with fully labeled data, except when the positive data set is seriously under-sampled. Our results also show that SVMC trains much faster than MC with very close accuracy.

3.4.1 Performance Measures

To evaluate a SCC method, we use the F_1 measure, which is a commonly used performance measure for SCC and also used [37] – one of the most recent works on SCC for text classification. This measure combines precision and recall in the following way:

$$precision = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ preditions}$$

$$recall = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ examples}$$

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

We also report the classification accuracy, even though it is not a good measure for the SCC problem because always predicting negative would give a high accuracy.

3.4.2 Experiment Methodology

To fully test the effectiveness and efficiency of SVMC, we compare seven methods – *MC, SVMC, OSVM, SVM_NN, S-EM, NB_NN, ISVM.*

- OSVM is described in Section 3.1.1.

- SVM_NN is standard SVM trained using positive data, with unlabeled data as a subsititute for negative data. This is not unreasonable, as the unlabeled data can be thought of as a good approximation of negative data. However, the small number of false positives are likely to affect the support vectors (SVs) of the boundary, which hurts the recall performance, as shown in our experiments.

- S-EM is described in Section 3.1.1.

- NB_NN (Naive Bayes with Noisy Negatives) also uses the unlabeled data as negative data.

- ISVM is the Ideal SVM trained from completely labeled data (P, with true N whi ch is manually classified from U). ISVM shows the ideal performance when the unlabeled data are labeled.

We used LIBSVM version 2.36[2] for SVM implementation of SVMC, MC, SVM_NN, OSVM, and ISVM. For the Reuters and breast cancer data sets, we used linear kernels for which are simple and accurate enough. (Text classification tasks are generally linearly separable [32].) For the letter recognition data set, we use Gaussian kernels which usually perform the best for pattern recognition problems.

For S-EM and NB_NN, we used the recent implementation released by the author of S-EM.[3].

Note that for a conventional SCC method, one cannot perform a validation process to optimize the parameters because no negative data is available. For SVMC, MC, SVM_NN, and ISVM, we used a theoretically motivated fixed parameter (e.g., $\nu = 0.01$ or 0.1). As discussed at the end of Section 3.2.3, a low setting (i.e., $\nu = 0.01$ or 0.1) of ν (the parameter to control the rate of noise in the training data) performed well since P is not likely noisy. (We used ν-SVM for the semantically meaningful parameter [44, 12].) For the Gaussian kernel parameter γ, we used the default value $\gamma = 1/m$ where m is the number of dimensions. The default γ performed very well on the letter recognition data set. However, kernel parameters such as γ usually affects the performance very much and thus need a careful optimization. Optimizing the kernel parameters for MC with only positive and unlabeled data is an important further work.

[2]http://www.csie.ntu.edu.tw/∼cjlin/libsvm
[3]http://www.cs.uic.edu/∼liub/S-EM/readme.html

We also used default parameters for S-EM and NB_NN. For OSVM, Tax proposed a sophisticated method to optimize the parameters without negative data [48]. However, their method is infeasibly inefficient especially in high dimensional spaces. As OSVM performs very poorly with default parameters, we linearly searched for the best parameter for OSVM based on the testing data set in order to simulate the results after the optimization. Note that the true performance of OSVM in practice would be poorer than those reported in our experiments since we unfairly performed the optimization only for OSVM.

3.4.3 Data sets

We used three data sets from three different domains: Reuters-21578[4] for text classification, and for pattern recognition and bioinformatics, the letter recognition and breast cancer data sets from the UCI machine learning repository[5].

We used the ModApte version of the Reuters corpus, which has 9603 training documents and 3299 testing documents. Each document was represented as a stemmed, TFIDF weighted word frequency vector.[6] Each vector had unit modulus. A list of stop words were removed, and words occurring in less than three documents were also ignored. Using this representation, the document vectors had around 10000 dimensions. We choose five frequently occurring categories for our experiments – *earn, grain, wheat, corn, crude*. Experiment on each category is performed independently.

The letter recognition data set includes 20000 samples of 26 capital letters, with each letter having about 800 samples. Each sample is represented by 16 numerical features of statistical moments and edge counts. We choose the first five letters for our experiments – *A, B, C, D, E*.

The breast cancer data set has 569 samples–*357 benign and 212 malignant*. Each sample is represented by 30 real-valued features that are the descriptions of each cell. We classify the malignant cells as positive class.

For each experiment, we divided the full positive set into two disjoint subsets p_1 and p_2. p_1 is to generate a set of positive examples P, while p_2 is to provide positive examples that can be added to U as noise. For instance, the ModApte split of the Reuters divides the entire set into two subsets – 9603 training and 3299 test documents. We set p_1 and p_2 to the positive data

[4]http://www.daviddlewis.com/resources/testcollections/reuters21578
[5]http://www.ics.uci.edu/ mlearn/MLRepository.html
[6]We used Rainbow (www-2.cs.cmu.edu/∼mccallum/bow/rainbow/) for text processing.

from the training and testing set respecti vely. For the letter recognition and breast cancer data sets, we set p_1 to two third of the full positive set and p_2 to the other one third.

We vary the amount of positive examples (controlled by α) as well as the amount of noise (i.e., positive examples) in the unlabeled data (controlled by β). Specifically, the positive example set P is composed of α fraction of p_1, and the unlabeled data U is composed of β fraction of p_2 and all the negative data. For example, if $\alpha = 1.0$ and $\beta = 0.5$, then 100% of p_1 are used as positive examples, and 50% of p_2 are added to U as noise. Performance is measured on U; we are essentially testing each method's ability of identifying the positive examples in U. Our experiment setup is very similar to that in [37], except our setup is more realistic: In [37], U is composed of β% of positives (e.g., class 'A') and samples from another class (e.g., class 'B'). Our U is β% of positives and the remainder is from all other classes. Note that ISVM trains from P, with true N manually classified from U, which implies that the same negative data are used for both the training and testing set in ISVM.

3.4.4 Results and Discussion

3.4.4.1 Overall results

Table 3.1 shows the performance results in realistic situations ($\alpha = \beta = 1.0$) where:

- for the Reuters, P is all the positive data in the training set, and U is all the negative data in the training set, with all data in the testing set.
- for the letter recognition data set, P is a randomly sampled two third of the full positive set, and U is the other one third, with a randomly sampled half of the full negative set. The results are averaged from five runs.
- for the breast cancer data set, P is a randomly sampled two third of the full positive set, and U is all others. The results are averaged from five runs.

We have the following observations from Table 3.1.

- MC and SVMC have the highest performance among all the methods trained from positive and unlabeled data. Although MC and SVMC show very similar performance, MC mostly performed a little better than SVMC as discussed in Section 3.3.2. The number of SVs of MC is also very close to that of ISVM while that of SVMC is a little smaller.

- SVMC always trained much faster than MC.[7] The difference of the training time between them could vary depending on the number of iterations they undergo.

- The optimized OSVM still performs worse than MC or SVMC due to the inability of using the unlabeled data for SVs. The number of SVs of OSVM is much smaller than others since they use only positives for SVs.

- SVM_NN gives very low F_1 scores due to the low recall. The small number of false positives in SVM_NN are likely to affect the SVs of the boundary, which hurts the recall performance badly.

- S-EM shows the same performance as NB_NN. As noted in [37], S-EM outperforms NB_NN only when the positive data form a significant fraction of the universal set. (i.e., when $|P_U|$ is significantly large compared to $|U|$.)

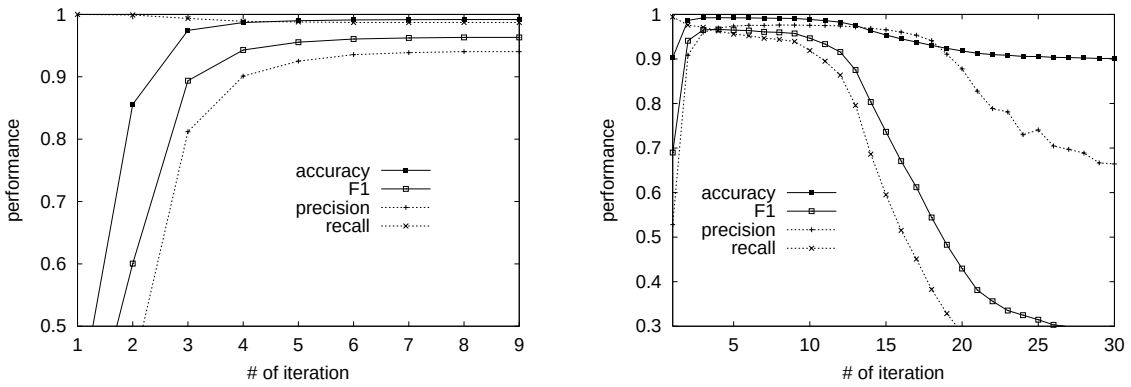
3.4.4.2 Results for different settings

Here we vary α and β to create different environments. Several observations can be made from Table 3.2:

- The performance of MC and SVMC drops abruptly when α is set very low. (In Table 3.2, the F_1 values are dropped significantly at $\alpha = 0.3$ for class "earn" in Reuters and $\alpha = 0.1$ for class "A" in the letter recognition.) The reason that the performance of MC and SVMC is susceptible to a low α is that, as we explicated via Theorem 3 in Section 3.2.3, when the positive training data are seriously under-sampled, the final boundary of MC and SVMC would trespass the true boundary and result in fitting around the under-sampled data. In this case, the intermediate boundaries of MC or SVMC generated in the middle of the iterations give better results. (See Figure 3.9(b).) We discuss this in more details in Section 3.4.4.3.

- β does not seem to influence the performance of MC and SVMC much as long as the positive data is not under-sampled, because the false negatives in U will be excluded in the training of the final boundary. Note that, since a different β essentially defines a different test set, the performance for different β is not really comparable. Thus, a

[7] We ran our experiments on a linux machine of Pentium III 800MHz with 384 MB memory.



(a) $\alpha = \beta = 1.0$ (b) $\alpha = 0.3$ and $\beta = 1.0$

Figure 3.9 SVMC performance convergence on the "earn" class

slightly low F_1 score of MC and SVMC for low β does not necessarily mean an inferior performance; it can be a result of an "easier" classification task due to lower amount of noise. The classification accuracy is very stable but not informative, since it is dominated by the large number of true negative examples in our test set (U).

- OSVM shows stable performance less affected by α or β. The stable performance is achieved by the unfairly performed optimization which is very hard to perform in high dimensional spaces. However, for low dimensional spaces, with the optimization method of [48], OSVM could give more stable performance that others when α is very low.

- As expected, SVM_NN suffers from very low recall performance, and almost always gives a nearly zero F_1 score.

- S-EM still shows the same performance as NB_NN, which implies that S-EM might not be useful for common positive classes as in our experiments. ([37] gives an example of "the class of movies we may enjoy watching" as a dominant positive class where S-EM might outperform NB_NN.)

3.4.4.3 Performance convergence of SVMC

Figure 3.9(a) shows the SVMC performance convergence on the "earn" class when the positive data is not seriously under-sampled. After the first iteration, the recall was very high, but the precision was very low because the initial class boundary covers a wide area including many false positives in the feature space. As it iterates, the precision and the overall

performance increases rapidly with a little decrement of the recall. Our experiments on all other classes in Table 3.1 also show similar shapes of the convergence graph.

Figure 3.9(b) shows the SVMC performance convergence when the positive data are too much under-sampled. The convergence happens after 30 iterations, but the highest performance is actually achieved at 5th iteration. As discussed in Section 3.2.3, MC obviously "over-iterated" in this case. Determining the best number of iterations is a hard problem because an SCC method is assumed to have no labeled negative data available for training or optimization, which makes impossible to use existing validation methods to determine the performance dropping point. How to improve SVMC's performance with very few positive data is clearly an important direction for future research.

3.5 Conclusions and further work

We study Single-Class Classification (SCC) with positive and unlabeled data, which is a fundamental problem in various domains, such as text classification or pattern recognition. We present the MC algorithm which computes an accurate classification boundary of the positive data without relying on labeled negative data by iteratively applying a margin maximization classifier, such as SVM, to progressively improved negative data from the unlabeled data. We study the SVMC algorithm that optimizes the MC algorithm for fast training by training with minimally required data sets in each iteration, which generates an accurate classification boundary in asymptotically the same time as SVM. We evaluate MC and SVMC using various domains of real data sets. Experiment results show that with a reasonable amount of positive data, the MC and SVMC algorithm give the best performance among all the existing SCC methods that we compared with. When the positive training data is seriously under-sampled or no wide gaps exist between positive and negative classes, the boundary of MC or SVMC tend to be over-conservative and much tighter than the true optimal boundary; in such a case, the intermediate boundaries of them before converging give much better results. It would be interesting to investigate an optimization technique to uncover the best number of iterations of SVMC given positive and unlabeled data.

Class	F_1 (1st row), accuracy (2nd row), # of SVs (3rd row)							t-time (sec.)	
	MC	SVMC	OSVM	SVM_NN	S-EM	NB_NN	ISVM	MC	SVMC
earn	0.9653 0.9923 (1209)	0.9632 0.9918 (1058)	0.8267 0.9578 (374)	0.0092 0.8921 (1948)	0.8327 0.9475 -	0.8327 0.9475 -	0.9893 0.9977 (1264)	963.87	119.62
grain	0.8239 0.9960 (636)	0.8211 0.9959 (493)	0.6962 0.9929 (143)	0.1250 0.9888 (788)	0.4307 0.9720 -	0.4307 0.9720 -	0.8905 0.9976 (642)	178.30	40.12
wheat	0.7737 0.9976 (349)	0.7571 0.9973 (259)	0.6621 0.9961 (63)	0.1519 0.9947 (441)	0.2190 0.9623 -	0.2190 0.9623 -	0.8413 0.9984 (370)	87.75	22.45
corn	0.7692 0.9979 (400)	0.7521 0.9977 (313)	0.4923 0.9948 (68)	0.2162 0.9961 (463)	0.1571 0.9595 -	0.1571 0.9595 -	0.8119 0.9985 (397)	78.02	22.19
crude	0.7308 0.9922 (689)	0.7111 0.9917 (527)	0.6685 0.9904 (121)	0.0309 0.9850 (851)	0.6119 0.9834 -	0.6119 0.9834 -	0.8724 0.9966 (656)	319.17	54.06
letter 'A'	0.9840 0.9991 (1385)	0.9840 0.9991 (1048)	0.8457 0.9922 (150)	0.0811 0.9726 (891)	0.4757 0.9467 -	0.4757 0.9467 -	0.9929 0.9996 (1407)	171.77	45.37
letter 'B'	0.9046 0.9950 (619)	0.9204 0.9959 (529)	0.7207 0.9869 (193)	0.0834 0.9758 (905)	0.1725 0.8489 -	0.1725 0.8489 -	0.9651 0.9983 (618)	75.61	14.34
letter 'C'	0.9641 0.9982 (1202)	0.9641 0.9982 (733)	0.7354 0.9882 (207)	0.0758 0.9755 (908)	0.1229 0.7293 -	0.1229 0.7293 -	0.9860 0.9923 (1205)	155.70	29.93
letter 'D'	0.9300 0.9963 (651)	0.9300 0.9963 (554)	0.6921 0.9860 (207)	0.0902 0.9752 (930)	0.1610 0.8881 -	0.1610 0.8881 -	0.9820 0.9991 (715)	80.47	16.06
letter 'E'	0.9419 0.9970 (698)	0.9396 0.9969 (567)	0.7112 0.9878 (186)	0.1333 0.9764 (885)	0.1861 0.8617 -	0.1861 0.8617 -	0.9798 0.9990 (709)	98.13	21.57
b-cancer	0.9470 0.9841 (56)	0.8444 0.9658 (23)	0.6315 0.8332 (12)	0.1872 0.2664 (116)	0.4741 0.6659 -	0.4741 0.6659 -	0.9521 0.9858 (55)	0.131	0.025

Table 3.1 Performance results ($\alpha = \beta = 1.0$). t-time: training time

Class	settings	F_1 (1st row), accuracy (2nd row)						
		MC	SVMC	OSVM	SVM_NN	S-EM	NB_NN	ISVM
"earn" in Reuters	$\alpha = 1.0, \beta = 0.7$	0.9513	0.9500	0.7768	0.0109	0.8120	0.8120	0.9931
		0.9923	0.9921	0.9595	0.9250	0.9709	0.9709	0.9990
	$\alpha = 1.0, \beta = 0.5$	0.9427	0.9381	0.7432	0.0069	0.8033	0.8033	0.9877
		0.9927	0.9921	0.9605	0.9398	0.9747	0.9747	0.9985
	$\alpha = 1.0, \beta = 0.3$	0.9017	0.9065	0.6426	0.0113	0.7491	0.7491	0.9914
		0.9919	0.9924	0.9656	0.9624	0.9779	0.9779	0.9994
	$\alpha = 0.7, \beta = 1.0$	0.9646	0.9625	0.8412	0.0073	0.8146	0.8146	0.9855
		0.9922	0.9917	0.9626	0.8920	0.9621	0.9621	0.9969
	$\alpha = 0.5, \beta = 1.0$	0.9639	0.9621	0.8468	0.0073	0.7900	0.7900	0.9822
		0.9921	0.9917	0.9674	0.8920	0.9583	0.9583	0.9962
	$\alpha = 0.3, \beta = 1.0$	0.3667	0.2836	0.8763	0.0037	0.7664	0.7664	0.9774
		0.9149	0.9012	0.9736	0.8918	0.9559	0.9559	0.9768
letter "A"	$\alpha = 1.0, \beta = 0.7$	0.9879	0.9679	0.0558	0.0826	0.6348	0.6348	0.9976
		0.9995	0.9987	0.9792	0.9795	0.9789	0.9789	0.9999
	$\alpha = 1.0, \beta = 0.5$	0.9721	0.9295	0.0305	0.0889	0.2828	0.2828	0.9922
		0.9923	0.9983	0.9871	0.9875	0.9428	0.9428	0.9998
	$\alpha = 1.0, \beta = 0.3$	0.9747	0.9342	0.0714	0.0200	0.1988	0.1988	0.9875
		0.9996	0.9990	0.9920	0.9926	0.9438	0.9438	0.9998
	$\alpha = 0.7, \beta = 1.0$	0.9627	0.9685	0.0833	0.0632	0.4612	0.4612	0.9721
		0.9980	0.9983	0.9732	0.9729	0.9436	0.9436	0.9985
	$\alpha = 0.5, \beta = 1.0$	0.9714	0.9577	0.0242	0.0811	0.6181	0.6181	0.9929
		0.9987	0.9981	0.9755	0.9726	0.9782	0.9782	0.9996
	$\alpha = 0.3, \beta = 1.0$	0.9278	0.9507	0.0242	0.0242	0.6615	0.6615	0.9394
		0.9967	0.9977	0.9755	0.9755	0.9823	0.9823	0.9972
	$\alpha = 0.1, \beta = 1.0$	0.3253	0.6065	0.0072	0.0072	0.6290	0.6290	0.8270
		0.9775	0.9842	0.9722	0.9722	0.9815	0.9815	0.9918

Table 3.2 Performance results for different settings. $|P_U|$: # of positives in U

CHAPTER 4

Discovering Compact and Highly Discriminative Feature Combinations via Support Vector Machines

4.1 Introduction

Due to the high throughput of modern experiment techniques, analyzing and modeling the massive biological data becomes increasingly important and has become a crucial step to derive useful biological knowledge. During the past a few years, many computational models have been built and deployed to perform functional classification and annotation. It is often the case that all potentially relevant features are collected and used to build the classification model. Such model is typically very complicated as it captures the dependency of the target function to features or feature combinations. It has been well known that only a subset of features may play determinant roles in the target function and that many collected features may be correlated. How to evaluate the importance of a feature or a feature combination with respect to the target function and how to discover the set of essential features has become an active research topic. This will lead to a deeper understanding of the system and to a more succinct model. Correlation mining and feature selection methods are two popular approaches to tackle these issues.

- **Correlation mining techniques** find most discriminative rules (or feature combinations) using a statistical metric (e.g., chi-square test). However, those rules are not very compact nor very discriminative since they do not take into account mutual information between feature or feature combinations, that is, they evaluate each rule independently. Also the complexity of those methods is fundamentally exponential to the number of features since considering feature combinations is a combinatorial problem.

- **Feature selection methods using support vector machine (SVM)** extract a highly discriminative feature set using a discriminative classifier (i.e., SVM). In terms of discrimination (or classification) quality, these methods are better than correlation mining methods based on statistical metrics. However, due to the fact that SVM model is "hidden", these methods do not disclose the relations within the feature set. We discuss related work in details in Section 4.2.

In this chapter, we focus on finding a compact set of features or feature combinations that mostly discriminates (or classifies) the target class. For example, in Figure 4.1(A), let d be the target class and the others are features. Then a set $\{\{\overline{f}\}, \{a, b\}\}$ is a most discriminative compact set of feature combinations. The set $\{\{\overline{f}\}, \{a, b\}, \{a, b, e\}\}$ is also discriminative but not *compact* since the feature combination $\{a, b, e\}$ is *redundant* of $\{a, b\}$.

The key ideas of our method are as follows:

1. We take advantage of SVM's kernel trick (polynomial kernel) to find $k'(<< k)$, where k is the number of features and k' is the minimum degree of feature combinations that needs to be considered to generate the highest classification performance. In other words, considering more than k' number of feature combinations would not generate higher classification performance but decrease the *generalization* performance by overfitting with too specific or redundant feature combinations. (An example of "redundant" feature sets is given at the end of Section 4.2.1.) As we will see in Section 4.3, SVM's kernel trick (polynomial kernel) allows us to find k' in linear time in terms of k.

2. After we generate an SVM model using the polynomial kernel of degree k', we extract the ranked features or feature combinations from the model, which would be much less time-consuming since we know k' that is much smaller than k.

3. To find a small subset of highly discriminative feature combinations, we can perform the recursive feature elimination of [25] but with the polynomial kernel of degree k'.

Our experiments in Section 4.5 show that our method produces better feature sets which give higher classification accuracy than the state-of-the-art feature selection method. In addition, our method provides more information – the relations between those feature sets.

a	b	c	d	e	f
1	1	1	1	1	0
1	1	0	1	1	0
1	0	1	0	1	1
1	0	0	0	1	1
0	1	1	0	1	1
0	1	0	0	1	1
0	0	1	1	1	0
0	0	0	1	1	0

(A) Examples of Transactions

I $\rightarrow$ C	support	confidence	correlated ?
$\{a\} \rightarrow \{b\}$	25%	50%	No
$\{a\} \rightarrow \{e\}$	50%	100%	No
$\{a, b\} \rightarrow \{d\}$	25%	100%	Yes
$\{a, b, e\} \rightarrow \{d\}$	25%	100%	Yes
$\{f\} \rightarrow \{d\}$	0%	0%	No
$\{\overline{f}\} \rightarrow \{d\}$	100%	100%	Yes
...	..	..	..

(B) Examples of Association Rules

Figure 4.1 Example of transactions and association rules

The remainder of this chapter is organized as follows. Section 4.2 gives an overview of research related to our work. The behavior of the SVM polynomial kernel and our method of discriminative feature combination discovery are presented in Sections 4.3 and 4.4, respectively. Section 4.5 shows the experimental study on drug activities and discuss the results. The conclusions are drawn in Section 4.6.

4.2 Related Work

4.2.1 Correlation mining techniques

For example of Figure 4.1(A), we want to find a set of features or feature combinations that discriminates a target feature d (or highly correlated the target class d). Each row represents a data record, and each column denotes a feature. "1" indicates the presence of the feature in the row, and "0" indicates the feature's absence.

Association rule mining

Let I be a feature set, and let $Pr(I)$ denote the ratio of the number of data records that include I to the number of all records. We call $Pr(I)$ the *support* of I. An association rule has the form $I_1 \rightarrow I_2$, where I_1 and I_2 re disjoint feature sets. The support of $I_1 \rightarrow I_2$ is defined as $Pr(I_1 \cap I_2)$, while the *confidence* is $Pr(I_1 \cap I_2)/Pr(I_1)$. For example, the support and the confidence of $\{a, b\} \rightarrow \{c\}$ are 12.5% and 50% respectively.

The support of a feature set is *anti-monotone* w.r.t. set-inclusion of feature sets; that is, for any $I \subseteq J, Pr(I) \geq Pr(J)$. Thus, whenever a feature set is not large w.r.t. a minimum support

threshold, neither is any of its supersets. The Apriori algorithm uses this heuristic to effectively prune away a substantial number of unproductive feature sets [2, 3]. There have been numerous further work of the association rule mining based on the support-confidence framework.

The support-confidence framework is weak at catching correlations between feature sets. For example, the first and second rules of Figure 4.1(B) statistically do not make sense because in each rule the assumptive feature set, say I, and the conclusive feature set (or target class), say C, are independent. On the other hand, in the third rule of Figure 4.1(B), the assumption and the target class are highly and positively correlated.

Correlation mining

Discovering correlation of feature sets which discriminates the target class typically calls for statistical measures such as chi-squared value, entropy information gain, gini index, or interclass variance. Correlation-based mining methods use such measures, say chi-squared value, for minimum threshold instead of using support and confidence.

The revision of the Apriori algorithm adopting the chi-squared test has been investigated [9]. One suffered from generating too many uncorrelated rules due to still using the support threshold [9]. S. Morishita suggested a scalable statistical pruning method by computing an upper bound of a statistical metric such as chi-squared value, but the upper bound of the statistical metric is only valid for binary feature set [40].

Correlation techniques have the following limitations:

- They are not scalable to the size of features: considering every possible combination of features is a combinatorial problem.
- They do not yield compact feature sets. For example, in Figure 4.1, although the third and fourth rules have the same correlation values, the fourth one ($\{a, b, e\} \rightarrow \{d\}$) is a redundant rule of the third rule ($\{a, b\} \rightarrow \{d\}$).
- Complementary feature sets that individually do not separate well the data are missed. Evaluating the discrimination power of an individual feature set does not take into account mutual information between feature sets.

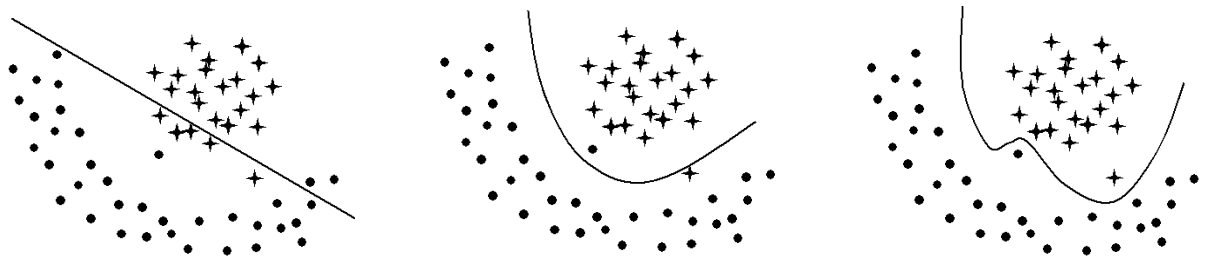
4.2.2 Feature selection methods using SVM

Many feature ranking algorithm using correlation coefficients have been explored with many applications to bioinformatics [17, 21, 20]. Recently, feature selection methods using SVM have been researched to find a small subset of features that maximize the classification performance.

The recursive feature elimination (RFE) algorithm for gene selection for cancer classification was proposed by I. Guyon et al. [25]. The RFE algorithm normalizes each feature value into the same range, train an SVM model from the normalized data, and use the weight of each feature as the indication for the importance of the feature. When the feature values are normalized well, the weight of each feature derived from the SVM model implies the contribution of the feature for discriminating the target class. However, their method is limited to a linear kernel which is often too restrictive to express a good discriminative function for many biological data. Also linear kernels do not take into account feature combinations so it can rank only individual features. Our method is similar to RFE in the sense that we also utilize the weight information to find a set of most discriminative feature but we discover feature combinations using SVM polynomial kernels.

J. Weston et al. have recently proposed a feature selection method for nonlinear kernels. Feature selection for nonlinear kernels is fundamentally a combinatorial problem which requires searching over all possible subsets of features. They use gradient descent method to approximate the results. However, the gradient descent method introduces another parameters, and improper setting of the parameter could cause jig-jag effect or slow convergence. Even when the parameter is optimized well, the converged point is still a local minimum. When the kernel is complex nonlinear and the number of features are large, there could exist numerous local minima in which the performance of the gradient descent method is strongly dependent on the randomly-set starting point. In our experiments, compact feature sets generated by our method show higher classification accuracy than the feature sets found by the local minimum search method. Moreover, the local minimum search methods cannot discover the relations between the features. That is, they are for data classification not for data interpretation.

We discover the compact subsets of highly discriminative features or feature combinations by exploring SVM polynomial kernels. Our method has higher discrimination (or classification)



(a) Underfitting (b) Desirable fitting (c) Overfitting

Figure 4.2 Example of boundaries overfitting and underfitting

power than the local minimum search methods, and also disclose the relations of the feature sets that is beneficial to data interpretation.

4.3 Behavior of SVM Polynomial Kernel

4.3.1 SVM Overview

In machine learning theory, the "optimal" class boundary function (or hypothesis) $h(x)$ given a limited number of training data set $\{(x, y)\}$ (y is the label of x) is considered the one that gives the best *generalization* performance which denotes the performance on "unseen" examples rather than on the training data. The performance on the training data is not regarded as a good evaluation measure for a hypothesis because the hypothesis ends up *overfitting* when it tries to fit the training data too hard. When a problem is easy to classify and the boundary function is complicated more than it needs to be, the boundary is likely overfit. When a problem is hard and the classifier is not powerful enough, the boundary becomes underfit. (Figure 4.2 shows examples of overfitting and underfitting.) SVM is an excellent example of supervised learning that tries to maximize the generalization by maximizing the *margin* and also supports nonlinear separation using advanced kernels, by which SVM tries to avoid overfitting and underfitting [10, 51, 13]. The *margin* in SVM denotes the distance from the boundary to the closest data in the feature space.

In SVM, the problem of computing a margin maximized boundary function is specified by the following quadratic programming (QP) problem:

$$minimize: \quad W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j)$$

$$subject\ to: \qquad \sum_{i=1}^{l} y_i \alpha_i = 0$$
$$\forall i : 0 \leq \alpha_i \leq C$$

The number of training data is denoted by l, α is a vector of l variables, where each component α_i corresponds to a training data (x_i, y_i); x_i is a vector representation of an example, and y_i is the class label of the example x_i. C is the soft margin parameter controlling the influence of the outliers (or noise) in training data.

The kernel $k(x_i, x_j)$ for linear boundary function is $x_i \cdot x_j$, a scalar product of two data points. The nonlinear transformation of the feature space is performed by replacing $k(x_i, x_j)$ with an advanced kernel, such as polynomial kernel $(x^T x_i + 1)^d$ or Gaussian kernel $e^{-g||x_i - x||^2}$. The use of an advanced kernel is an attractive computational short-cut, which foregoes an expensive creation of a complicated feature space. An advanced kernel is a function that operates on the input data but has the effect of computing the scalar product of their images in a usually much higher-dimensional feature space (or even an infinite-dimensional space), which allows one to work implicitly with hyperplanes in such highly complex spaces.

4.3.2 Notes on SVM kernels

SVM polynomial kernel and Gaussian kernel have been widely used in pattern recognition due to its high expressible power [48, 8, 35]. Especially, Gaussian kernel has an infinite VC dimension,[1] which implies that it is basically able to draw any kinds of boundary functions. (e.g., In [35], Gaussian kernel draws a checkerboard shape of boundary.) However, it is very hard to extract the feature weights from Gaussian kernel because it is very difficult to express the kernel as an inner product of two $\Phi(x)$ functions [13].

Polynomial kernel on the other hand also has a higher expressible power than linear kernel[2] by conceptually considering feature combinations up to d degree whereas linear kernel only takes into account each feature independently. (We describe the behavior of this polynomial kernel in details in the next section.) It is also possible to extract the feature weights from the polynomial kernel because it is possible to express the kernel as an inner product of two $\Phi(x)$

[1] In machine learning, the expressible power of a classification function type is often measured by VC dimension, and SVM Gaussian kernel $e^{-g||x_i - x||^2}$ is proven to have infinite VC dimension [10].

[2] The VC dimension of polynomial kernel is $\binom{m+d-1}{d}$ whereas that of linear kernel is $2m+1$, where m is # of dimensions in the feature space and d is the degree of polynomial kernel [10].

functions. (An example of extracting the $\Phi(x)$ function from a polynomial kernel is shown in Chapter 6 of [27], and we describe the algorithm of extracting feature weights from a polynomial kernel in Section 4.4.)

4.3.3 Behavior of Polynomial Kernel

A polynomial kernel on feature set $\mathcal{F}$ generates the same classification results as a linear kernel on feature set $\mathcal{F}'$–the combinations of each feature in $\mathcal{F}$–without an explicit creation of the feature set $\mathcal{F}'$. The polynomial kernel has a parameter d which denotes the degree of feature combinations. For instance, SVM with the polynomial kernel of $d = 3$ computes the function $f(x)$ formulated as:

$$f(x) = \sum_{i}^{n} w_i x_i + \sum_{i}^{n}\sum_{j}^{n} w_{ij} x_i x_j + \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{n} w_{ijk} x_i x_j x_k, \tag{4.1}$$

where x_i is the ith feature of an example x and w_i is the weight of the feature x_i. The classification function (1) takes into account the combinations of every feature up to the third degree. For instance, when $d = 2$, a two-dimensional feature space of feature set $\{x_1, x_2\}$ will be transformed into a higher-dimensional space that includes also the second degree of each feature – $\{x_1, x_2, x_1x_2, x_1^2, x_2^2\}$.

The boundary function becomes more powerful and complicated as d increases, and at some point the boundary function starts overfitting and degrading its generalization performance. Figure 4.2(a) shows an example of underfitting when d is too low. Figure 4.2(c) is a case of overfitting when d is too high. Overfitting due to high d corresponds to generating redundant feature sets in the correlation mining of Section 4.2.1. The d showing the peak performance is different depending on the data set.

4.4 Feature Combination Discovery (FCD) Algorithm

Considering all the combinations of features to discover a discriminative subset of feature combinations fundamentally takes an exponential time in terms of the number of features. However, SVM with a polynomial kernel is able to draw a classification function that considers d degree of feature combinations in linear time regardless of d. The key idea of our method is first to find the d of best generalization performance. (The best d is usually small in biological

data sets. In all our experiments, $d \leq 3$.) Then we compute the weight of each feature combination up to d degree from the SVM model and thus avoid the expensive and fruitless search of higher degrees of combinations. The computed feature combinations will be compact and highly discriminative.

Figure 4.3 describes the FCD algorithm which extracts the most compact subset of feature combinations that generates higher classification accuracy than the user threshold θ. The first part of the algorithm – Steps 1 and 2 – finds the degree d of feature combinations showing the peak performance by using the SVM polynomial kernel. As we discussed in Section 4.3, a polynomial kernel with d on a feature set $\mathcal{F}$ generates the same effects as a linear kernel on d degree of feature combinations over $\mathcal{F}$. As d increases, the classification function starts overfitting and degrading the performance at some point, at which we stop the iteration. In our experiments in Section 4.5, the performance start degrading after $d = 2$.

There are several ways to estimate the generalization performance of SVM such as cross validation or estimation of leave-one-out errors. Cross validation techniques have been used for estimating the generalization error of a decision rule. However, it is computationally too expensive to perform iteratively in our framework. Estimating the SVM generalization performance has been researched actively based on a strong theoretical foundation. One of the most recent methods, estimation of leave-one-out errors [31], has shown to be efficient and fairly accurate. We used this method in our experiments.

Given the generated SVM model of d degree polynomial kernel, Steps 3 to 5 extracts the most compact subsets of feature combinations that generates higher classification accuracy than the user threshold θ as follows:

- Step 4.1. computes the weight of each feature combination. Figure 4.4 describes how to extract the weights from an SVM model of polynomial kernel with degree d. First a hash table H is initialized whose key will be an attribute combination and the value will be the weight of the attribute combination. The key of the combination of more than d degree will be excluded. An SVM model is a collection of (x_i, c_i), where c_i is the non-zero coefficient of an example x_i, which corresponds to $y_i\alpha_i$ in the QP formula. Only support vectors will have non-zero c_i. The weight of a feature combination A will be determined

by the support vectors that includes the elements of A. For instance, if $A = a_1 a_2$, then the weight of A is $\sum_i c_i a_1 a_2$, where i is all the support vectors that have both a_1 and a_2.

- Step 4.2. removes the least discriminative δ number of features from the previous feature set F by taking the highest ranked $len(F) - \delta$ number of features. This removing step is equal to that of recursive feature elimination in [25]. For computational reasons, it may be more efficient to remove several features at a time at the expense of possible classification performance degradation. With the lower δ, it is likely to generate the more compact set [25]. Our experiments in Section 4.5 also show that a more compact set generated 100% accuracy with $\delta = 1$ or 5 than with $\delta = 10$.

- Steps 4.3. and 4.4. re-construct the training set with the reduced feature set and train another SVM model from the training set.

- As iterating Step 4., the classification performance may start decreasing at some point, and we return the feature set before the performance becomes lower than the user threshold. Our experiments show that the classification performance stays stable until the point that the performance starts decreasing rapidly (See Figure 4.5).

We discuss the computational complexity of our algorithm at the end of Section 4.5.

4.5 Experimental Evaluations

In this section, we experimentally evaluate our method (the FCD algorithm). To evaluate the discrimination quality and the compactness of the feature combinations derived b the FCD algorithm, we compare the classification performance of the derived feature set with the state-of-the-art feature selection method [54]. The relations within the subset are disclosed as a by-product in the FCD algorithm.

We use a dataset of 294 chemical compounds. A total of 264 features are collected. Each feature is a numerical descriptor that represents a property of the compound structure, such as mass, diameter, etc. The target classification is on whether the compound is an active drug.

We used LIBSVM version 2.36[3] for the SVM implementation. We used ν-SVM because of its semantically meaningful parameter. The soft margin parameter ν in ν-SVM denotes

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvm

ZM1V, SNar, Xt, RHyDp, Jhetp, X0, X5A, X0v, X1v, X0Av, X2Av, X3Av, PW3, PW4, PW5, PJI2, CSI, ECC, MDDD, UNIP, IDE, IDM, IDET, HVcpx, Uindex, SIC0, BIC0, SIC1, BIC1, IC2, SIC2, BIC2, IC3, SIC3, BIC3, IC4, BIC4, IC5, BIC5, LP1, SEigv, SEige, SEigp, AEigm, VRA1, VRZ2, VRm2, D_Dr07, D_Dr11, T_N_S_

Table 4.1 The 50 discriminative feature derived by FCD

the upper-bound of noise rate in training data set [12]. We fixed $\nu = 0.01$; the classification accuracy is insensitive to the parameter when it is set in a reasonable range and the data set does not have a significant amount of noise.

The SVM with linear kernel showed less than 30% accuracy on this data set, which implies that the data set is not linearly separable and thus considering only each indivisual feature is not enough for classifying the drug activity. The RFE method thus will not generate good results.

When we use SVM with polynomial kernel, it showed 100% accuracy with $d \geq 2$, which implies that the drug activity is only characterized by combinations of features. Thus, we run the recursive elimination with the polynomial kernel with $d = 2$, and it generated the results shown in Figure 4.5.

As we discussed in Section 4.2.2, the classification performance of the local minimum search method is unstable because the performance depends on the starting point and there exists numerous local minimum points when the SVM involves a large number of features with nonlinear kernels. In Figure 4.5, FCD with $\delta = 1$ or 5 shows 100% accuracy with 50 features whereas the local minimum search method requires at least 80 features to achieve 100% accuracy. (We report the average accuracy of five runs of their method.) The feature sets generated by the local minimum search show lower classification accuracy than those generated by FCD at each number of features.

Table 4.5 shows the 50 features that the FCD (with $\delta = 5$) extracts, and Table 4.5 shows within the 50 features the top 10 positive and negative feature combinations respectively sorted by their absolute weights. It is obvious that the best descriptor combinations are not necessarily composed of the most important individual descriptors.

Using higher degree of polynomial kernel could allow to extract even more compact features – less than 50 features – which generates 100% accuracy. For instance, in our experiments,

Positively Discriminative Combination	Negatively Discriminative Combination
IDET · SIC1 (15.99)	UNIP · IDET (-11.88)
ECC · HVcpx (14.08)	X0v · X2Av (-10.98)
PW4 · HVcpx (12.41)	ECC · SEigp (-10.79)
ECC · ECC (10.81)	RHyDp · ECC (-10.64)
PJI2 · ECC (10.59)	IDET · BIC0 (-10.38)
ECC · IDM (10.34)	IDE · IDE (-10.34)
UNIP · LP1 (9.86)	ECC · AEigm (-9.96)
IDET · BIC1 (9.84)	ECC · Uindex (-9.91)
CSI · IC3 (9.79)	CSI · ECC (-9.89)
Xt · CSI (9.63)	X0v · X3Av (-9.74)
...	...

Table 4.2 Positively and negatively discriminative feature combinations and their weights

SVM with $d = 3$ reduced into 30 the number of features needed to generate 100% accuracy. (See Figure 4.6.) However, as we discussed in Section 4.3, the generalization performance of the function would degrade. The 50 features we extracted here with $d = 2$ can be viewed as the most compact set that generates the same classification and generalization performance as the original data set which also generates the best performance with $d = 2$.

Computational Considerations

The FCD algorithm runs SVM iteratively whose training time is quadratic to the number of data set (n) and linear to the number of features (m). Our experiments indicated that better features are obtained with smaller δ in the iteration of FCD. However, there are only significant differences for the smaller subset of features (e.g., less than 100), which suggests that, without trading accuracy for speed, one can remove chunks of features at the beginning of the iterations and remove smaller number of features as it iterates. Then, we can approximate the number of total iteration into $log(m)$. Extracting the weights from a polynomial kernel with d degree takes m^d time. Thus the complexity of FCD becomes $O(n^2 * m^{d+1} * log(m))$ if the complexity for training an SVM is $O(n^2 * m)$.

In our experiments on the 294 data records of 264 features, using the LIBSVM, training an SVM with polynomial kernel with $d = 2$ takes less than two seconds in a Pentium III 800Mhz machine. Running the FCD on the data set took about two hours in the same environment. However, it is quite acceptable in bioinformatics that the data analysis takes a few hours because the data collection and preparation usually take much longer, e.g., several months or years.

When the data set needs a high degree of combinations, our current method could become extremely slow because, as we see from the complexity, the time for extracting the weights from an SVM model has a exponential dependency to the degree of the combinations. Approximating the weights for fast extraction could be an interesting further research for this feature selection problem.

4.6 Conclusions

In this chapter, we present a new method that uses SVM weight information to extract discriminative feature combinations from a rich feature space. We have shown that this method delivers a promising result on the drug activity data that has a large number of features and a relatively small number of instances. It successfully identifies a much smaller set of features that produce a better classification quality than the (larger) set of features selected by other methods. As one direction of our future work, we shall evaluate the trend of the classification quality as the number of features increases and develop a more systematic solution to determine the form of kernel functions. We also plan to deploy the proposed method on other biological data such as the gene expression data and the protein subcellular location data.

Input: - Training data set $X = [(x_1, y_1), ..., (x_n, y_n)]$
- Discrimination (classification accuracy) threshold θ

Output: - A compact set of feature combinations F

Algorithm:

/* Find d of the best performance */

1. Initialize $d = 1$
2. Repeat
   2.1. $M = train\_SVM(X, d)$
   2.2. estimate the performance of M
   2.3. if the performance is estimated lower than that of previous loop, then $d = d - 1$ and exit the loop
   2.5. increase d by 1

/* Extract a compact set of feature combinations whose classification accuracy is higher than the threshold θ */

3. Initialize F = the feature set
4. Repeat
   4.1. $H = compute\_weight(M, d)$ /* See Figure 4.4 */
   4.2. $F' = return\_top\_feature(H, len(F) - \delta)$
   4.3. $X = rebuild(X, F')$
   4.4. $M = train\_SVM(X, d)$
   4.5. if $test(M)$ is lower than θ, then exit the loop
   4.6. $F = F'$
5. Return F

Figure 4.3 Description of the Feature Combination Discovery (FCD) algorithm

Input: - Support vectors and corresponding coefficients
$S = [(x_1, c_1), (x_2, c_2), ..., (x_n, c_n)]$
- Degree d

Output: - A hash table H of a ranked list of feature combinations

Algorithm:

1. Initialize a hash table H.
   /* each key is an attribute combination and the value is the weight of the attribute combination */
2. Repeat for each support vector $x_i = (a_1, a_2, ..., a_m)$
   2.1. Repeat for each attribute combination A whose number of combination is equal to or less than d
      2.1.1. Multiple the coefficient c_i of s_i to the value of A and accumulate the results into the hash table $H[A]$

/* At this point, the hash table H includes the key-value pairs of every feature combination less than d degree and the corresponding weight of the attribute */

3. Sort the hash table H by its values
4. Return H

Figure 4.4 Description of the function "*compute_weight*(M, d)" that is called in the loop of the FCD algorithm

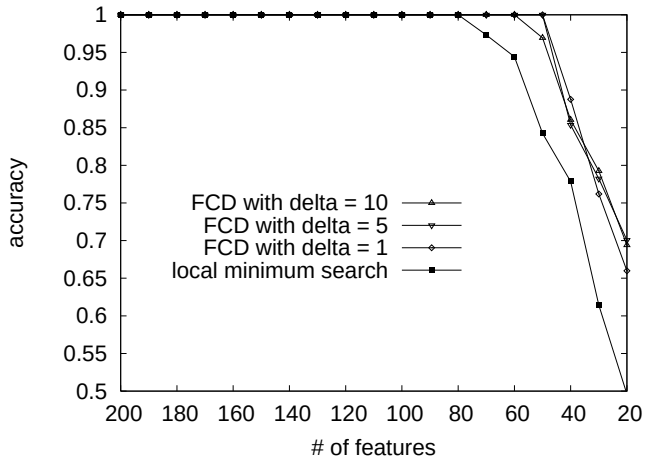


Figure 4.5 Comparison of classification performance between FCD and the local minimum search method when $d = 2$. FCD generates 100% accuracy with 50 and 60 features while the local minimum search method does it with 80 features.

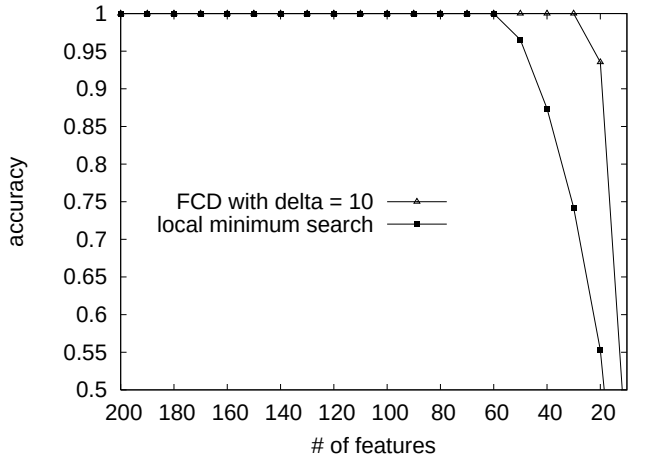


Figure 4.6 Comparison of classification performance between FCD and the local minimum search method when $d = 3$. FCD generates 100% accuracy with 30 features while the local minimum search method does it with 60 features.

CHAPTER 5

Conclusions and Future Work

Data mining has been an important methodology to extract useful knowledge from a variety of data sources such as Web & text documents, relational databases, and biological data. In machine learning community, Support Vector Machines (SVMs) has been favorably used for classification and regression problems because of their salient properties such as margin maximization and nonlinear classification using kernel tricks. Despite the SVM's prominence, there have been some critical obstacles to enjoy such advantages of SVMs in large-scale data mining applications. Mainly, this is because (1) SVMs are not very scalable to large data sets, (2) they are limited to binary classification, and (3) the generated models of SVMs are hard to understand.

In this thesis, these challenges are identified and the SVM technologies are advanced to the practice of data mining with the proposal of several practical data mining solutions. The following questions are addressed in the thesis:

- **How to make SVMs scalable to build classification functions from large data sets?** SVMs are known to be infeasibly slow and exert too much system resources, e.g., memory, when training large data sets. Many real-world data mining applications involve millions or billions of data records such that often multiple scans of the entire data are too expensive to perform nor feasible is running an SVM with limited amount of system resources. It becomes an important problem to make SVMs scalable and enable their prominent properties on large data sets with limited amount of system resources. Chapter 2 tries to answer this question by proposing a new method *CB-SVM* which efficiently trains large data sets with high accuracy given limited amount of system resources.

- **How to build an accurate classification function from only positive and unlabeled data when acquiring a reasonable sample of negatively labeled data is hard or expensive?** This problem is called Single-Class Classification (SCC). SCC problems are prevalent in real-world, e.g., text and Web classification, pattern recognition, and bioinformatics. Chapter 3 proposes a new method *SVMC* that draws an accurate classification boundary from positive and unlabeled data.

- **How to identify a much smaller set of features or feature combinations that produce as good classification quality as the (larger) set of features?** This question has become a crucial issue in bioinformatics where the feature space is very high due to the rich information about functionalities of certain biological entities. Typically, among such rich features, a small number of features or feature combinations play determinant roles in functional discriminations such as classification and prediction. Chapter 4 proposes a new method that employs the SVMs as the classification means and aim at finding compact feature combinations.

5.1 Summary

In Chapter 2, we presented *Clustering-Based SVM (CB-SVM)*, a specifically designed SVM classification method for handling very large data sets [57]. CB-SVM applies a hierarchical micro-clustering algorithm that scans the entire data set only once to provide an SVM with high quality samples that carry the statistical summaries of the data such that the summaries maximize the benefit of learning the SVM. CB-SVM tries to maximize the SVM performance for very large data sets given a limited amount of resources. Our analyses show that the training complexity of CB-SVM quadratically depends on the number of support vectors which is much less than that of entire data. In our experiments, we ran CB-SVM on the network intrusion detection data set from the KDD Cup 1999, which consists of about five million training data records so that it would take too long to use existing SVMs to train a classification function from it. Basically, CB-SVM is more effective on very large data sets having different probability distributions of training and testing data on which random sampling usually hurts. The network intrusion data set is a good example of those because the testing data is not from the same probability distribution as the training data, and it also includes specific attack types not in

the training data. This is because they were collected in different times of periods, which makes the task more realistic. CB-SVM efficiently trained an accurate classification function which classifies the abnormal data packets.

In Chapter 3, we presented two single-class classification (SCC) methods, MC (Mapping Convergence) and SVMC (Support Vector Mapping Convergence), that employ SVM's margin-maximization property to compute an accurate classification boundary from only positive and unlabeled data (without negatively labeled data) [55]. In SCC problems, it is assumed that a reasonable sample of the negative data is not available. Since it is not natural to collect the "non-interesting" objects (i.e., negative data) to train the concept of the "interesting" objects (i.e., positive data), SCC problems are prevalent in real world where positive and unlabeled data are widely available but negative data are hard or expensive to acquire. The basic idea of our method is to exploit the natural "gap" between positive and negative data by incrementally labeling negative data from the unlabeled data using the *margin maximization* property of SVM. In our experiments on the data sets of text documents, letter recognition, and breast cancer classification, MC and SVMC without labeled negative data significantly outperformed other SCC methods and generated as accurate boundaries as standard SVM with fully labeled data when the positive data is not very under-sampled. Our results also show that SVMC trains much faster than MC with very close accuracy.

Another challenging problem in using SVMs for data mining is to understand the SVM model for further analysis and mining knowledge from the data. Chapter 4 presents a method that analyzes the SVM model to extract useful information, with an application of it in bioinformatics [58]. Nowadays, high throughput techniques such as microarray experiments make it feasible to examine and collect massive data at the molecular level. These data, typically mapped to a very high dimensional feature space, carry rich information about functionalities of certain biological entities and can be used to infer valuable knowledge for the purposes of classification and prediction. Typically, a small number of features or feature combinations play determinant roles in functional discrimination. The identification of such features or feature combinations is of great importance. In this chapter, we study the problem of discovering compact and highly discriminative features or feature combinations from a rich feature collection. We employ the support vector machine as the classification means and aim at finding compact feature combinations. Comparing to previous methods on feature selection, which identify fea-

tures solely based on their individual roles in the classification, our method is able to identify minimal feature combinations that ultimately have determinant roles in a systematic fashion. Our experimental study on drug activity data shows that the method can discover descriptors that are not necessarily significant individually but are most significant combinatively.

5.2 Future Work

Improvement on SVM's scalability and kernel tricks of SVM is still an active on-going research issue. Many related research communities (e.g., bioinformatics, text and Web, computer vision, and natural language processing) have increasingly adopted SVMs and try to keep up with the SVM's newest technology for their own problems. This thesis plays a role as a step in the direction towards applying and developing the SVM-based methodology for practical data mining problems. The followings are several interesting directions of further work on this thesis.

- **Developing an efficient high dimensional index structure for SVMs.** The CB-SVM algorithm does not perform well on high dimensional data because the hierarchical micro-clusters, the index structure to support SVM, become not much meaningful in high dimensional spaces [59]. (Eclidean distance metric becomes not much meaningful in high dimensional spaces [7].) Developing an effective high dimensional indexing structure for SVMs is clearly a crucial problem since the efficient processing of high dimensional data is one distinctive strength of the original SVMs. Clustering or indexing and search techniques for high dimensional data spaces have been actively researched [53, 4, 5, 6], which gives an implication on the strong feasibility in this direction of research.

- **Developing more accurate stopping criteria that make the boundary less susceptible to overfitting.** The boundary of MC or SVMC tend to be over-conservative and much tighter than the true optimal boundary when the positive training data is seriously under-sampled or no wide gaps exist between positive and negative classes in the feature space; in such a case, the intermediate boundaries of them before converging give much better results. Determining the best number of iterations is a hard problem because an SCC method is assumed to have no labeled negative data available for training

or optimization, which makes impossible to use existing validation methods to determine the performance dropping point. It is an important direction for future research to investigate an optimization technique to uncover the best number of iterations of SVMC given positive and unlabeled data.

- **Investigate the SVM-based feature extraction methodology for other applications of bioinformatics.** In bioinformatics, due to the high throughput techniques such as microarray experiments, it becomes feasible to examine and collect massive data at the molecular level, which makes the data typically map to a very high dimensional feature space, carry rich information about functionalities of certain biological entities and can be used to infer valuable knowledge for the purposes of classification and prediction. Typically, a small number of features or feature combinations play determinant roles in functional discrimination. The identification of such features or feature combinations is of great importance. In Chapter 4, we developed an SVM-based method that discovers compact and highly discriminative features or feature combinations from a rich feature collection, and we applied it to a drug activity data. In our future work, we shall deploy the proposed method on other biological data such as the gene expression data and the protein subcellular location data.

Beyond the above problems, there are numerous data mining sub-tasks and problems, for which SVMs can be exploited and applied to provide principled and consolidating solutions, which produce higher quality results with far less human intervention.

REFERENCES

[1] D. K. Agarwal. Shrinkage estimator generalizations of proximal support vector machines. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'02)*, pages 173–182, 2002.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data (SIGMOD'93)*, pages 207–216, 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. Int. Conf. Very Large Databases (VLDB'94)*, pages 487–499, 1994.

[4] S. Berchtold, C. Bohm, H. V. Jagadish, H.-P. Kriegel, and J. Sander. Independent quantization: An index compression technique for high-dimensional data spaces. In *Proc. Int. Conf. Data Engineering (ICDE'00)*, pages 577–588, 2000.

[5] S. Berchtold, B. Ertl, D. A. Keim, H.-P. Kriegel, and T. Seidl. Fast nearest neighbor search in high-dimensional spaces. In *Proc. Int. Conf. Data Engineering (ICDE'98)*, pages 209–218, 1998.

[6] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree: An index structure for high-dimensional data. In *Proc. Int. Conf. Very Large Databases (VLDB'96)*, pages 28–39, 1996.

[7] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proc. Int. Conf. Database Theory*, pages 217–235, 1999.

[8] S. M. Bileschi and B. Heisele. Advances in component-based face detection. In *IEEE Int. Workshop on Analysis and Modeling of Faces and Gestures*, pages 149–156, 2003.

[9] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. In *Proc. ACM SIGMOD Int. Conf. Management of Data (SIGMOD'97)*, pages 265–276, 1997.

[10] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[11] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Proc. Advances in Neural Information Processing Systems (NIPS'00)*, pages 409–415, 2000.

[12] C.-C. Chang and C.-J. Lin. Training nu-support vector classifiers: theory and algorithms. *Neural Computation*, 13:2119–2147, 2001.

[13] N. Christianini and J. Shawe-Taylor. *An Introduction to support vector machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[14] Ronan Collobert and Samy Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[15] F. DeComite, F. Denis, and R. Gilleron. Positive and unlabeled examples help learning. In *Proc. Int. Conf. Algorithmic Learning Theory (ALT'99)*, pages 219–230, 1999.

[16] F. Denis. PAC learning from positive statistical queries. In *Proc. Int. Conf. Algorithmic Learning Theory (ALT'99)*, pages 112–126, 1998.

[17] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[18] A. Frosini, M. Gori, and P. Priami. A neural network-based model for paper currency recognition and verification. *IEEE Transactions on Neural Networks*, 7(6):1482–1490, November 1996.

[19] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'01)*, pages 77–86, 2001.

[20] T. S. Furey, N. Christianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Hauessler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[21] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[22] M. Gori, L. Lastrucci, and G. Soda. Autoassociator-based models for speaker verification. *Pattern Recognition Letters*, 17:241–250, 1995.

[23] Russell Greiner, Adam J. Grove, and Dan Roth. Learning active classifiers. In *Proc. Int. Conf. Machine Learning (ICML'96)*, pages 207–215, 1996.

[24] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 73–84, 1998.

[25] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[26] J. Han and M. Kamber. *Data Mining: concepts and techniques.* Morgan Kaufmann, 2001.

[27] S. Haykin. *Neural Networks A Comprehensive Foundation.* Prentice Hall, 1999.

[28] O. Watanabe J. L. Balczar, Y. Dai. A random sampling technique for training support vector machines. In *Int. Conf. Data Mining (ICDM'01)*, pages 43–50, 2001.

[29] T. Joachims. Making large-scale support vector machine learning practical. In A.J. Smola B. Scholkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines.* MIT Press, Cambridge, MA, 1998.

[30] T. Joachims. Text categorization with support vector machines. In *Proc. European Conf. Machine Learning (ECML'98)*, pages 137–142, 1998.

[31] T. Joachims. Estimating the generalization performance of a SVM efficiently. In *Proc. Int. Conf. Machine Learning (ICML'00)*, pages 431–438, 2000.

[32] T. Joachims. A statistical learning model of text classification with support vector machines. In *Proc. ACM SIGIR Int. Conf. Information Retrieval (SIGIR'01)*, pages 128–136, 2001.

[33] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[34] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. In *Proc. Advances in Neural Information Processing Systems (NIPS'01)*, pages 785–792, 2001.

[35] Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *SIAM Int. Conf. Data Mining*, 2001.

[36] F. Letouzey, F. Denis, and R. Gilleron. Learning from positive and unlabeled examples. In *Proc. Int. Conf. Algorithmic Learning Theory (ALT'00)*, pages 11–30, 2000.

[37] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *Proc. Int. Conf. Machine Learning (ICML'02)*, pages 387–394, 2002.

[38] L. M. Manevitz and M. Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[39] O. L. Mangasarian and D. R. Musicant. Active support vector machine classification. Technical report, Computer Sciences Department, University of Wisconsin at Madison, 2000.

[40] S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *Proc. ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS'00)*, pages 226–236, 2000.

[41] J. Platt. Fast training of support vector machines using sequential minimal optimization. In A.J. Smola B. Scholkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[42] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proc. Int. Conf. Machine Learning (ICML'00)*, pages 839–846, 2000.

[43] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[44] B. Scholkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1083–1121, 2000.

[45] B. Scholkopf, R.C. Williamson, A.J. Smola, and J. Shawe-Taylor. SV estimation of a distribution's support. In *Proc. Advances in Neural Information Processing Systems (NIPS'00)*, pages 582–588, 2000.

[46] A. J. Smola and B. Scholkopf. A tutorial on support vector regression. Technical report, NeuroCOLT2 Technical Report NC2-TR-1998-030, 1998.

[47] N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines. In *Proc. the Workshop on Support Vector Machines at the Int. Joint Conf. on Articial Intelligence (IJCAI'99)*, 1999.

[48] D. M. J. Tax and R. P. W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal of Machine Learning Research*, 2:155–173, 2001.

[49] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proc. Int. Conf. Machine Learning (ICML'00)*, pages 999–1006, 2000.

[50] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[51] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[52] W. Wang, J. Yang, and R. R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proc. Int. Conf. Very Large Databases (VLDB'97)*, pages 186–195, 1997.

[53] W. Wang, J. Yang, and R. R. Muntz. PK-tree: A spatial index structure for high dimensional point data. In *Proc. Int. Conf. Foundations of Data Organization FODO'98*, pages 281–293, 1998.

[54] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Proc. Advances in Neural Information Processing Systems (NIPS'00)*, pages 668–674, 2000.

[55] H. Yu. SVMC: Single-class classification with support vector machines. In *Proc. Int. Joint Conf. on Articial Intelligence (IJCAI'03)*, pages 567–572, 2003.

[56] H. Yu, J. Han, and K. C. Chang. PEBL: Positive-example based learning for Web page classification using SVM. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'02)*, pages 239–248, 2002.

[57] H. Yu, J. Yang, and J. Han. Classifying large data sets using SVMs with hierarchical clusters. In *Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'03)*, pages 306–315, 2003.

[58] H. Yu, J. Yang, W. Wang, and J. Han. Discovering compact and highly discriminative features or feature combinations of drug activities using support vector machines. In *IEEE Computer Society Bioinformatics Conf. (CSB'03)*, pages 220–228, 2003.

[59] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pages 103–114, 1996.

VITA

Hwanjo Yu was born in Seoul, Korea on September 24, 1974. He graduated from the Chung-Ang University at Seoul in 1997 with a degree in computer science and engineering. From his junior in the college he started working as a system developer for Sunware network marketing company in Seoul and in Los Angeles before relocating to Champaign, Illinois in 1998, to purdue graduate study in computer science. During his graduate study, he received 2002 and 2003 SIGKDD student scholarship awards, the 2003 UIUC data mining gold award, a 2003 AAAI student scholarship award, and a 2003 CIKM student scholarship award. Following the completion of his Ph.D. in May, 2004, he will begin work for the computer science department in the university of IOWA at Iowa city as an assistant professor.